TUGAS AKHIR – KI141502

RANCANG BANGUN SISTEM *ENTERPRISE RESOURCE PLANNING* PADA MODUL *INVENTORY AND WAREHOUSE MANAGEMENT* BERORIENTASIKAN *MULTI-TENANCY* DENGAN *DISTRIBUTED DATABASE*

I Gede Arya Putra Perdana
NRP 5112 100 151

Dosen Pembimbing I
Dwi Sunaryono, S.Kom., M.Kom.

Dosen Pembimbing II
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

RANCANG BANGUN SISTEM *ENTERPRISE RESOURCE PLANNING* PADA MODUL *INVENTORY AND WAREHOUSE MANAGEMENT* BERORIENTASIKAN *MULTI-TENANCY* DENGAN *DISTRIBUTED DATABASE*

I Gede Arya Putra Perdana
NRP 5112 100 151

Dosen Pembimbing I
Dwi Sunaryono, S.Kom., M.Kom.

Dosen Pembimbing II
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

DESIGN AND IMPLEMENTATION OF ENTERPRISE RESOURCE PLANNING IN INVENTORY AND WAREHOUSE MANAGEMENT WITH MULTI-TENANCY ORIENTED USING DISTRIBUTED DATABASE

I Gede Arya Putra Perdana
NRP 5112 100 151

Supervisor I
Dwi Sunaryono, S.Kom., M.Kom.

Supervisor II
Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM *ENTERPRISE RESOURCE PLANNING* PADA MODUL *INVENTORY AND WAREHOUSE MANAGEMENT* BERORIENTASIKAN *MULTI-TENANCY* DENGAN *DISTRIBUTED DATABASE*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
I GEDE ARYA PUTRA PERDANA
NRP. 5112 100 151

Disetujui oleh Dosen Pembimbing Tugas Akhir:

DWI SUNARYONO, S.Kom., M.Kom. ..... (pembimbing 1)
NIP: 19720528 199702 1 001

Prof. Drs. Ec. Ir. RIYANARTO SARNO, M.Sc., Ph.D. ..... (pembimbing 2)
NIP: 19590803 198601 1 001

SURABAYA
JULI, 2016

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN SISTEM *ENTERPRISE RESOURCE PLANNING* PADA MODUL *INVENTORY AND WAREHOUSE MANAGEMENT* BERORIENTASIKAN *MULTI-TENANCY* DENGAN *DISTRIBUTED DATABASE*

Nama	**: I Gede Arya Putra Perdana**
NRP	**: 5112100151**
Jurusan	**: Teknik Informatika – FTIf ITS**
Dosen Pembimbing I	**: Dwi Sunaryono, S.Kom., M.Kom.**
Dosen Pembimbing II	**: Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.**

Abstrak

ERP atau Enterprise Resource Planning adalah sebuah sistem informasi yang digunakan untuk mengintegrasikan dan mengotomasikan proses bisnis yang berhubungan pada aspek operasi, produksi maupun distribusi pada sebuah perusahaan. Salah satu modul yang merupakan bagian utama dan kunci dari modul ERP yaitu Supply Chain Management dan telah dikerjakan pada tugas akhir ini yaitu Warehouse Management System. Inventory and Warehouse Management bertujuan untuk mengontrol segala proses pergerakan barang yang terjadi seperti pengiriman (shipping), penerimaan (receiving), penyimpanan (put away), pergerakan (move), dan pengambilan (picking).

Dalam tugas akhir ini, pada proses Warehouse Management menggunakan konsep FIFO (First In First Out) dalam penentuan barang yang akan di proses lebih lanjut untuk menghindari berkurangnya kualitas dari barang tersebut. Metode FIFO mengambil catatan barang terlama pada sistem dan menggunakan barang tersebut untuk proses berikutnya. Sistem pencatatan utama yang telah dikerjakan pada tugas akhir ini adalah pencatatan barang diluar gudang (Transfer Posting), pencatatan perpindahan barang didalam gudang (Warehouse Movement), pencatatan barang datang (Good Receipt),

pencatatan permintaan barang (Good Issue) dan pencatatan data informasi barang (Item Master Data).

Hasil implementasi menunjukkan bahwa proses pencatatan barang menggunakan konsep FIFO dan proses bisnis yang terdapat pada Inventory and Warehouse Management berhasil diimplementasikan.

Kata kunci: Inventory and Warehouse Management, Enterprise Resource Planning, IWM, ERP, Multi-Tenancy, Distributed Database, FIFO

DESIGN AND IMPLEMENTATION OF ENTERPRISE RESOURCE PLANNING IN INVENTORY AND WAREHOUSE MANAGEMENT WITH MULTI-TENANCY ORIENTED USING DISTRIBUTED DATABASE

Student Name	**: I Gede Arya Putra Perdana**
NRP	**: 5112100151**
Major	**: Informatics Department – FTIf ITS**
Supervisor I	**: Dwi Sunaryono, S.Kom., M.Kom.**
Supervisor II	**: Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc.,Ph.D.**

Abstract

ERP or Enterprise Resource Planning is an information system that used for integration and automation of business process that which are related to operation, production, nor distribution in a company. One module of ERP is Supply Chain Management and has been done on this final project is Warehouse Management System. Inventory and Warehouse Management intend to controlling all the good's movement process likes shipping, receiving, put away, move, and picking.

In this final project, the process of Warehouse Management uses the concept of FIFO (First In First Out) in the determination of the goods which will be processed further in order to avoid degrading the quality of the goods. FIFO took the oldest item records on the system and use of these items to the next process. The main recording system that has been done in this final project is the recording of goods outside the warehouse (Transfer Posting), recording the movement of goods within the warehouse (Warehouse Movement), recording the receive goods (Good Receipt), recording the demand for goods (Good Issue) and recording of data information items (Item Master Data).

The implementation results show that the process of recording the goods using the concept of FIFO and business processes contained in Inventory and Warehouse Management successfully implemented.

Keywords: Inventory and Warehouse Management, Enterprise Resource Planning, IWM, ERP, Multi-Tenancy, Distributed Database, FIFO

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pengerjaan Tugas Akhir, dan sistematika penulisan Tugas Akhir.

1.1 Latar Belakang

Warehouse atau pergudangan merupakan sebuah tempat atau area yang berfungsi menyimpan barang untuk / hasil produksi dalam jumlah dan rentang waktu tertentu yang kemudian didistribusikan ke lokasi yang dituju berdasarkan permintaan. Kendala yang dialami dalam pengelolaan *warehouse* adalah akurasi pergerakan barang dan menghitung rentang waktu barang disimpan. Melakukan *Warehouse Management* sangat ditekankan terutama pada sisi penempatan barang di dalam gudang. Proses penempatan barang yang tidak beraturan akan berakibat pada semakin lamanya *picking process*, rendahnya *inventory accuracy*, *dead stock*, *over stock*, dan *under stock.* Dibutuhkan kontrol aktivitas pergerakan barang dan dokumen untuk meningkatkan efisiensi penggunaan *warehouse* agar jumlah dan rentang waktu barang disimpan dalam nilai minimum atau sesuai perencanaan. Sistem ERP dengan *multi-tenancy* akan menambah kemudahan dalam penggunaan sistem ke beberapa perusahaan yang ingin menggunakan sistem ini secara bersamaan sehingga semakin banyak perusahaan yang dapat mengelola gudang dengan lebih baik.

Pada pengembangan sebelumnya, dikembangkan sebuah program *Inventory* yang meliputi *requesting*, *delivering*, *controlling stock*, serta *master data*. Aplikasi ini menggunakan arsitektur berorientasi *servi*ce (SOA) dengan

sistem *Model-View-Controller* (MVC) dan *Workflow* untuk .NET.

Untuk Tugas Akhir ini, lebih mendetailkan informasi-informasi pergerakan barang pada *Inventory* (*warehouse movement*). Modul IWM (*inventory and warehouse management*) dikembangkan untuk lebih mendetailkan informasi tersebut dimana meliputi submodul *good issue, transfer posting, good receipt, warehouse storage, warehouse movement* dan *master data*. Aplikasi ini menggunakan arsitektur *multitenancy* dengan sistem *Model-View-Controller* (MVC) dan *Workflow* untuk PHP. Pada mekanisme *Multi-Tenancy Separated Database*, penyimpanan data setiap *tenant* dilakukan secara *separated schema* dari *tenant* lainnya.

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana memodelkan *business process model* dan *data model* yang meliputi submodul *good issue, transfer posting, good receipt, warehouse storage, warehouse movement* dan *master data* terkait pada modul *inventory and warehouse management* (IWM) yang terdapat pada SAP ERP?
2. Bagaimana mengimplementasikan konsep FIFO (*First In First Out*) ke dalam sistem IWM ERP?
3. Bagaimana menangani kegagalan sistem pada sebuah *database server* sehingga kinerja *tenant* tidak terganggu?
4. Bagaimana mengaplikasikan *replication* pada *distributed database* terkait pada modul IWM ERP?

1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Hasil dari tugas akhir ini adalah sebuah aplikasi ERP *Inventory and Warehouse Management* yang terdiri dari submodul-submodul *good issue, transfer posting, good receipt, warehouse storage, warehouse movement* dan *master data.*
2. Model yang akan diintegrasikan prosesnya adalah *Inventory and Warehouse Management*.
3. Bahasa pemrograman yang akan digunakan adalah PHP.
4. *Platform* yang digunakan adalah Yii 2.0.
5. Menggunakan 5 buah komputer dimana 1 komputer untuk pengembangan aplikasi, satu computer untuk pengujian aplikasi, satu computer untuk manajemen *server* replikasi *database* dan 2 komputer sebagai *database server*
6. Sistem basis data yang digunakan adalah MySQL.
7. Barang *defect* pada sistem diasumsikan sebagai barang yang tidak dapat di-*disassembly* dan akan di *default*-kan tidak digunakan dalam sistem.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Mampu memodelkan *business process model* dan *data model* yang meliputi submodul *good issue, transfer posting, good receipt, warehouse storage, warehouse movement* dan *master data* terkait pada modul *inventory and warehouse management* (IWM) yang terdapat pada SAP ERP.
2. Mampu mengimplementasikan konsep FIFO ke dalam sistem IWM ERP.
3. Mampu menangani kegagalan sistem pada sebuah *database server* sehingga kinerja *tenant* tidak terganggu.
4. Mampu mengaplikasikan *replication* pada *distributed database* terkait pada modul IWM ERP.

1.5 Manfaat

Tugas akhir ini diharapkan dapat menyediakan sistem pengelolaan gudang sesuai dengan SAP ERP yang dapat digunakan oleh banyak perusahaan dalam satu waktu sehingga mampu menyediakan data pergerakan barang yang dapat digunakan untuk pengelolaan dan manajemen gudang yang lebih baik untuk meningkatkan kualitas perusahaan.

1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1. **Studi literatur**

   Studi literatur yang dilakukan berfokus pada pembuatan aplikasi PHP dengan *framework* Yii 2.0 sesuai dengan proses bisnis model dari ERP, implementasi *multi-tenancy* serta proses bisnis yang akan digunakan seperti *Warehouse Management System.*

2. **Analisis dan Perancangan Sistem**

   Aktor yang menjadi pelaku adalah pengguna perangkat lunak yang dibangun oleh penulis. Kemudian beberapa kebutuhan fungsional dari sistem ini adalah sebagai berikut:

   a. Analisis aktor yang terlibat didalam sistem.
   b. Perancangan diagram *use case*, yang merupakan analisis kebutuhan pada aplikasi yang akan dibangun.
   c. Analisis kebutuhan non-fungsional.
   d. Perancangan sistem informasi ERP.
   e. Perancangan fitur *Multi-Tenancy*, *Distributed Database*, RBAC.

3. **Implementasi**

   Pada tahap ini dilakukan pembuatan elemen perangkat lunak yang merupakan implementasi dari *workflow* IWM ERP menggunakan konsep FIFO.

4. **Pengujian dan evaluasi**

   Pada tahap ini dilakukan pengujian terhadap elemen perangkat lunak dengan menggunakan data uji

yang telah dipersiapkan. Pengujian dan evaluasi perangkat lunak dilakukan untuk mengevaluasi jalannya perangkat lunak, mengevaluasi fitur utama, mengevaluasi fitur-fitur tambahan, mencari kesalahan yang timbul pada saat perangkat lunak aktif, dan mengadakan perbaikan jika ada kekurangan.

5. **Penyusunan buku Tugas Akhir**

   Pada tahap ini dilakukan pendokumentasian dan pelaporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan Tugas Akhir.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan yang menjadi dasar dari pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan alur, proses dan perancangan antarmuka pada perangkat lunak.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak perangkat lunak dan implementasi fitur-fitur penunjang perangkat lunak.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian subjektif untuk mengetahui penilaian aspek kegunaan (*usability*) dari perangkat lunak dan pengujian fungsionalitas yang dibuat dengan memperhatikan keluaran yang dihasilkan serta evaluasi terhadap fitur-fitur perangkat lunak.

Bab VI Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir.

2.1 Penelitian Terkait

Tugas Akhir ini merupakan pengembangan riset berkelanjutan dari riset atau penelitian tentang ERP. Pada Tugas Akhir yang sebelumnya, modul IWM masih sangat sederhana dan belum melakukan pendataan gudang secara mendetail [1]. Pengembangan kali ini lebih mendetailkan sebuah modul IWM untuk menjalankan proses bisnis *Make to Order* dan *Make to Stock*. Selain itu pada modul IWM ini telah diberikan sebuah metode FIFO (*First In, First Out*) dimana bertujuan untuk mendata proses masuk barang menggunakan konsep *queue* dalam penataan barang dimana barang yang pertama masuk ke dalam gudang adalah barang pertama yang akan digunakan disaat terdapat permintaan barang.

Berikut perbandingan dari modul yang diberikan pada ERP sebelum dan ERP yang akan dikembangkan :

Tabel 2.1 Modul-modul ERP 2013 dan ERP 2016

No.	Nama Modul ERP 2013	Nama Modul ERP 2016
1	Account Payable & Receivable	Account Payable & Receivable
2	General Ledger	General Ledger
3	Cost Accounting	Finance
4	Customer Relationship Management	Sales and Distribution
5	Production Planning	Production
6	Supplier Relationship Management	Purchasing

No.	Nama Modul ERP 2013	Nama Modul ERP 2016
7	Inventory	Inventory and Warehouse Management
8	-	Assets Management
9	-	**Human Resource Management**

Tabel 2.2 Perbandingan Modul ERP 2013 dan ERP 2016 pada modul Inventory

ERP 2013	ERP 2016
Good Receipt	Good Receipt
Good Issue	Good Issue
Item Master	Transfer Posting
Warehouse Master	Warehouse Movement
	Item Master
	Warehouse Master

2.2 *Enterprise Resource Planning (ERP)*

ERP atau Perencanaan Sumber Daya Perusahaan merupakan suatu sistem terintegrasi yang terdiri dari berbagai macam modul proses bisnis perusahaan pada umumnya. [2] Sistem ERP disebut sebagai suatu sistem yang saling terkait dikarenakan penggunaan perangkat lunak sebagai sarana pengintegrasian antar bagian yang diinginkan untuk diintegrasikan. Sistem ERP ini seringkali digunakan perusahaan untuk mengelola data serta meneliti data-data krusial perusahaan yang tersebar di berbagai area bisnisnya, semisal data-data dari bagian keuangan, pemasaran, produksi, sumber daya manusia, manajerial, dan sebagainya. Sistem ERP juga memfasilitasi integrasi aliran data antar departemen yang terhubung dalam sistem tersebut.

2.2.1 **Proses Bisnis Umum ERP**

Simha R. Magal dan Jeffrey Word (2011) menyatakan terdapat beberapa proses bisnis utama yang terdapat pada ERP yang berfungsi sebagai aliran aplikasi ERP dalam sebuah

perusahaan. Beberapa proses binis utama, antara lain pengadaan (*procurement*), produksi (*production*), penjualan (*sales/fullfillment*). [3]

2.2.1.1 Proses Bisnis Pengadaan (procurement)

Proses bisnis pengadaan merupakan proses dimana suatu perusahaan membeli barang untuk diproduksi. Proses bisnis ini merupakan awal dalam proses *Make-to-Stock*, sebuah proses dimana perusahaan membuat barang sebelum adanya pembelian. Proses business ini dimulai dengan perusahaan membuat dokumen *purchase requisition*. Kemudian dokumen *purchase requisition* ini setelah disetujui akan menjadi dokumen *purchase order* atau perintah pembelian barang. Setelah barang sudah dibeli, maka muncul dokumen *receive material* yang menandakan barang sudah diterima. Kemudian perusahaan mendapatkan tagihan atau *invoice* untuk dibayar. Terakhir perusahaan membayarkan sejumlah uang untuk melunasi tagihan tersebut. Proses bisnis *procurement* ditunjukkan pada Gambar 2.1.

Gambar 2.1. Proses Bisnis Pengadaan Barang

2.2.1.2 Proses Bisnis Produksi (production)

Proses bisnis produksi merupakan proses perusahaan memproduksi barang. Untuk membuat suatu barang, perusahaan harus mengikuti proses bisnis ini. Proses bisnis ini dimulai dengan perusahaan membuat dokumen *request production*. Setelah

produksi disetujui maka akan muncul dokumen *authorized production*. Produksi akan dapat berjalan ketika bahan mentah sudah tersedia dan dapat dikeluarkan dari gudang. Proses produksi dimulai ketika dokumen *create product* sudah muncul. Setelah barang sudah jadi maka, barang akan dipindahkan dari tempat produksi ke gudang dengan adanya dokumen *receive finished goods*. Proses bisnis produksi ditunjukkan pada Gambar 2.2.

Gambar 2.2. Proses Bisnis Produksi

2.2.1.3 Proses bisnis penjualan (fullfillment)

Proses bisnis penjualan merupakan proses perusahaan menjual barang kepada pelanggan. Proses ini merupakan proses terpenting dalam simulasi bisnis. Proses bisnis ini dimulai dengan perusahaan membuat dokumen *sales order* untuk mengelola siapa yang membeli barang, barang yang dibeli, jumlah barang, harga barang, dan total harga. Kemudian proses dilanjutkan dengan mempersiapkan pengiriman barang. Barang akan dikirim dan muncul dokumen *shipment*. Setelah barang diterima oleh *customer*, maka perusahaan mengirimkan tagihan atau *invoice* kepada pelanggan. Proses bisnis ini berakhir ketika pelanggan sudah membayar tagihan tersebut (Magal & Word, Integrated Business Processes with ERP Systems, 2012). Proses bisnis penjualan ditunjukkan pada Gambar 2.4.

Gambar 2.3. Proses Bisnis Penjualan

2.3 *Warehouse Management System (WMS)*

Warehouse Management System (WMS) adalah sistem yang membantu *stakeholder* terkait pengelolaan pergerakan barang dari/ke dalam gudang sehingga dapat mempercepat proses *lead time* secara otomatis, mengetahui semua transaksi *inventory* dan jumlah *stock* lebih cepat dan akurat secara *real time*, dapat mengatur lokasi penyimpanan barang secara optimal, serta dapat melakukan alur distibusi barang dengan baik.

Simha R. Magal dan Jeffrey Word (2011) menyatakan bahwa *Inventory and Warehouse Management* (IWM) terdiri dari banyak proses yang terkait dengan pengelolaan barang yang terdapat di gudang. Contoh proses IWM adalah *Goods Receipt, Goods Issue, Transfer Postings, Stock Transfers*. Kemudian sebagai dasar acuan detail proses di dalam modul HCM. Kami mengacu dalam aplikasi ERP (Odoo, Adempiere, dan InoERP). Terdapat 4 proses bisnis pergerakan utama yang terdapat pada *inventory* [3] [5] [6], yakni :

2.3.1 *Goods Issue*

Good Issue didefinisikan sebagai pergerakan keluar barang secara fisik atau material dari gudang atau *Good Issue* adalah isu tentang barang fisik atau material dari gudang. Hal ini menghasilkan penurunan persediaan dari gudang. Sebagai contoh barang dikeluarkan dari toko ke departemen produksi untuk membuat produk.

2.3.2 *Goods Receipt*

Good Receipt didefinisikan sebagai pergerakan masuk barang secara fisik atau material ke gudang atau *good receipt* adalah penerimaan material di toko atau penerimaan persediaan dari *vendor* atau produsen. Semua *good receipt* menghasilkan bertambahnya persediaan material di gudang. Sebagai contoh diterimanya produk yang dihasilkan ke toko.

2.3.3 *Transfer Postings*

Transfer posting adalah penghapusan materi dari lokasi salah satu penyimpanan dan mentransfer ke lokasi penyimpanan lain. *Transfer posting* dapat terjadi baik dalam *plant* yang sama atau antara dua *plant* yang berbeda.

2.3.4 *Stock Transfers*

Stock Transfer adalah perpindahan fisik suatu barang dari satu lokasi penyimpanan ke lokasi penyimpanan lainnya. *Stock transfer* di sistem *Material Management* antara lain termasuk perpindahan fisik material dari suatu *plant* ke *plant* yang lain dan *storage location* ke *storage location* yang lain.

Untuk dapat menjalankan proses pengaturan i*nventory* dengan baik, dibutuhkan sebuah metode yang mampu memproses pencatatan barang untuk mengurangi terjadinya *dead stock, over stock,* dan *under stock.*

2.4 *FIFO (First In First Out)*

Metode FIFO menganggap bahwa harga pokok dari barang-barang yang pertama kali dibeli akan merupakan barang yang dijual pertama kali. Dalam metode ini persediaan akhir dinilai dengan harga pokok pembelian yang paling akhir. [7]

Metode ini juga mengasumsikan bahwa barang yang terjual karena pesanan adalah barang yang mereka beli. Oleh karenanya, barang-barang yang dibeli pertama kali adalah barang-barang pertama yang dijual dan barang-barang sisa di tangan (persediaan

akhir) diasumsikan untuk biaya akhir. [8] Karenanya, untuk penentuan pendapatan, biaya-biaya sebelumnya dicocokkan dengan pendapatan dan biaya-biaya yang baru digunakan untuk penilaian laporan neraca. [9]

Metode ini konsisten dengan arus biaya aktual, sejak pemilik barang dagang mencoba untuk menjual persediaan lama pertama kali. FIFO merupakan metode yang paling luas digunakan dalam penilaian persediaan. [10]

Metode FIFO seringkali tidak nampak secara langsung pada aliran fisik dari barang tersebut karena pengambilan barang dari gudang lebih didasarkan pada pengaturan barangnya. Dengan demikian meode FIFO lebih nampak pada perhitungan harga pokok barang. Dalam metode FIFO, biaya yang digunakan untuk membeli barang pertama kali akan dikenali sebagai *Cost of Goods Sold* (COGS). Untuk perhitungan harga maka digunakan harga dari stok barang dari transaksi yang terdahulu. [11]

a. Metode FIFO (*First In First Out*) pertama kali dikenal dalam akuntansi keuangan sebagai salah satu metode dalam penilaian persediaan barang. Harga yang digunakan sebagai dasar dalam menilai persediaan barang dapat memakai harga lama atau harga baru.
b. Pada metode FIFO, persediaan barang yang dikeluarkan untuk produksi atau dijual, nilainya didasarkan pada harga menurut urutan yang pertama masuk. Jadi, untuk penilaian pada persediaan barang yang tersisa, berarti harganya didasarkan pada harga baru atau harga urutan yang terakhir.

2.5 *Multitenant Data Architecture* atau *Multitenancy*

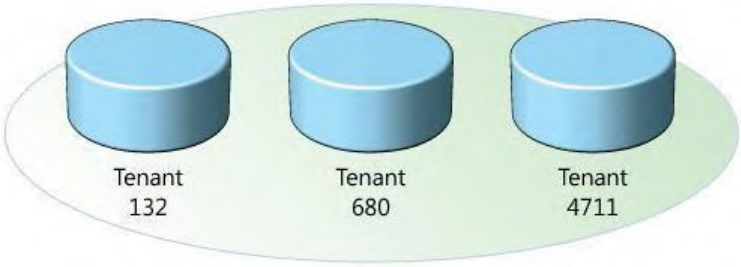
Multitenancy mengacu pada arsitektur perangkat lunak di mana salah satu contoh dari perangkat lunak yang berjalan pada server dan menyajikan beberapa *tenant*. *Tenant* adalah sekelompok pengguna yang berbagi akses umum dengan hak akses khusus untuk contoh perangkat lunak. Dengan arsitektur *multitenant*, aplikasi perangkat lunak ini dirancang untuk memberikan setiap tenant sebuah bagian untuk berdedikasi contoh termasuk data,

konfigurasi, manajemen pengguna, fungsi individu tenant dan sifat non-fungsional. *Multitenancy* bertentangan dengan arsitektur *multi-instance* misalnya, di mana contoh perangkat lunak terpisah beroperasi atas nama penyewa yang berbeda. [12]

Arsitektur data *Multitenancy* digunakan karena dinilai memadai dan cukup aman dalam mengatasi masalah kurang kepercayaan *tenant* untuk menyerahkan kontrol data bisnis *tenant* kepada pihak ketiga. Terdapat 3 jenis *Multitenancy*, antara lain: [13]

2.5.1 **Separated Database**

Separated Database adalah data setiap *tenant* disimpan pada *database* yang terpisah dengan *tenant* lain. Keuntungan arsitektur ini adalah mudah untuk mengatur kembali model data aplikasi yang digunakan. Tetapi memerlukan biaya yang cukup tinggi untuk menjaga peralatan *server* dan juga *back up* data dari setiap *tenant*. Model arsitektur *Separated Database* ditunjukkan pada Gambar 2.2.1**.**

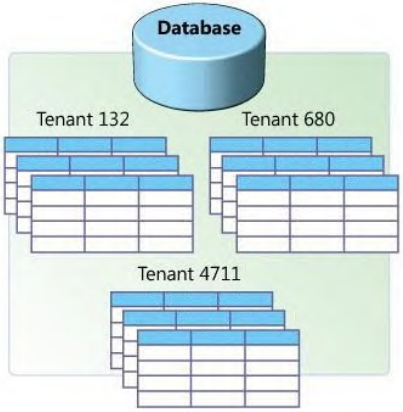


Gambar 2.4Separated Database

2.5.2 **Shared Database, Separated Schema**

Shared Database, Separated Schema data setiap *tenant* disimpan pada satu *database* tetapi pada *schema* yang terpisah dengan *tenant* lain. Keuntungan dari arsitektur ini adalah mudah untuk digunakan karena tabel yang dibentuk pada

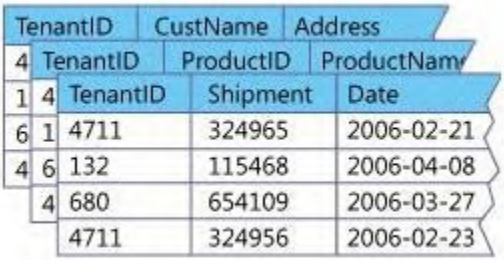
awalnya merupakan tabel standar, dan selanjutnya dapat diubah sesuai keinginan tenant. Akan tetapi apabila terjadi kegagalan maka perlu dilakukan perbaikan untuk semua tabel yang ada dalam *database*. Model arsitektur *Shared Database, Separated Schema* ditunjukkan pada Gambar 2.2.2



Gambar 2.5 Shared Database, Separated Schema

2.5.3 Shared Database, Shared Schema

Shared Database, Shared Schema adalah data setiap *tenant* disimpan pada satu *database* dan satu *schema*. Keuntungan dari arsitektur ini adalah tidak memerlukan biaya yang tinggi, akan tetapi apabila terjadi kegagalan maka perlu dilakukan perbaikan untuk semua tabel yang ada dalam *database*. Model arsitektur *Shared Database, Shared Schema* ditunjukkan pada Gambar 2.2.3**.**



Gambar 2.6 ***Shared Database, Shared Schema***

2.6 RBAC (*Role Based Access Control*)

RBAC adalah sistem yang diterapkan pada aplikasi yang berhubungan dengan pengontrolan akses sumber daya. RBAC memberikan hak akses untuk peran (roles). Perancang kebijakan atau administrator sangat berperan dalam memberikan hak kepada para pelaku, sehingga subjek akan mendapatkan akses ke objek melalui role yang telah diberikan oleh administrator (Khayat et al, 2005). Fitur yang disediakan oleh RBAC menjadi daya tarik bagi para perusahaan yang menerapkan teknologi informasi, pertama RBAC yang mengorganisir subjek dan role secara alamiah sesuai dengan struktur yang diterapkan pada perusahaan tersebut (Ferraiolo et al, 1997). Hubungan antara hak akses dengan para pelaku yang diterapkan pada perusahaan tersebut, pertama RBAC memberikan tugas keamanan pada kontrol akses sebagai prioritas tertinggi untuk mengontrol akses ke sumber daya. Hal tersebut mengakibatkan RBAC akan menerapkan keamanan yang sangat ketat dalam melakukan kontrol akses ke sumber daya. Kedua RBAC dalam menerapkan hak akses kepada pengguna membutuhkan waktu yang singkat, dengan cara menghubungkan subjek dengan *role*, sehingga memerlukan penunggasan hak akses untuk *role* pada setiap subjek (K hayat et al, 2005).

Kontrol akses dalam mengambil keputusan ditentukan oleh *role*, sehingga pengguna sebagai bagian dari sebuah organisasi akan mendapatkan hak akses sesuai dengan *role* yang didapatkannya. Kebijakan yang d ilakukan oleh RBAC akan

membuat control akses yang didapatkan oleh pengguna berdasarkan keputusan yang diperoleh dalam sebuah organisasi. Pengguna tidak bisa mengambil hak akses pengguna lain, hal inilah dasar perbedaan antara RBAC dan DAC (Ferraioloet al, 1992).

Dalam menangani kebijakan keamanan, RBAC adalah fondasi utama yang harus didefinisikan dengan baik sebelum *Security Constraint Specifiers* (SCS) atau disebut juga sebagai petugas yang menentukan kendala dalam menjaga keamanan.

SCS ini akan menentukan keterbatasan yang terjadi seperti kendala dalam penerapan sistem. RBAC memiliki banyak variasi yang akan dilakukan dengan menggambarkan bagaimana menentukan keterbatasan yang terjadi tanpa kehilangan ciri khasnya. Struktur RBAC meliputi *roles*, *permission*, *user*, dan *session* (C hen et al, 1996).

Dalam RBAC, role didefinisikan sebagai suatu gagasan yang merupakan dasar dari kebijakan kontrol akses (Sandhu et al, 1999). Pendefinisian *role* masih menjadi perdebatan sehingga diperlukan menjelasan konsep roleitu tersendiri, sehingga untuk melakukan kolaborasi dengan kontrol akses masih disesuaikan dengan kebutuhan. Dalam ilmu prilaku, *role* didefinisikan sebagai pola yang ditentukan sesuai prilaku seseorang dalam situasi tertentu berdasarkan posisi orang tersebut (Hawkins et al, 1983). Dengan kata lain pendefinian *role* adalah tugas yang didapatkan oleh seseorang sesuai dengan tanggung jawabnya. Namun demikian, sangat sulit untuk menggambarkan pengertian rolekarena bahasa yang diterapkan masih bersifat ambigu (Zhu, 2003).

2.7 Basis Data Terdistribusi (BDT)

Basis data terdistribusi adalah suatu basis data yang memiliki kontrol terpusat pada *server* manejemen dengan distribusi penyimpanan data yang tersebar di beberapa *server* data [8]. BDT diterapkan dalam rangka untuk menjaga prinsip *high availability* pada aplikasi ERP. BDT memungkinkan aplikasi tetap berjalan jika terdapat sebuah *server* basis data mengalami *system*

failure. Terdapat 2 mekanisme utama dalam BDT, yaitu replikasi dan fragmentasi.

2.7.1 Replikasi

Replikasi adalah mekanisme penyalinan seluruh atau sebagian tabel basis data ke beberapa *server data* [8]. Setiap transaksi (penambahan, penghapusan, atau perubahan data) akan dieksekusi pada semua *server* data yang menyusun sistem BDT. Hal ini menyebabkan mekanisme replikasi membebani performa sistem, namun memiliki tingkat kompleksitas yang paling sederhana.

2.7.2 Fragmentasi

Fragmentasi adalah mekanisme penyalinan sebagian data atau struktur dari setiap tabel basis data [8]. Transaksi basis data yang terjadi harus diolah terlebih dahulu oleh *server* manajemen untuk menentukan letak *server* data dari tabel yang akan terpengaruh. Hal ini menyebabkan mekanisme fragmentasi memiliki tingkat kompleksitas yang tinggi, namun memiliki performa yang lebih baik dibandingkan dengan mekanisme replikasi.

2.8 *Business Process Model and Notation* (BPMN)

BPMN adalah representasi grafis untuk menentukan proses bisnis dalam suatu pemodelan proses Bisnis. Tujuan utama dari BPMN adalah menyediakan suatu notasi standar yang mudah dipahami oleh semua pemangku kepentingan bisnis [14]. Dari analisis bisnis yang menciptakan draft permulaan dari proses-proses sampai dengan pengembang-pengembang teknis yang bertanggung jawab untuk mengimplementasikan teknologi yang membantu pelaksanaan proses. [15]

2.9 *Data Model*

Data Model adalah Sekumpulan konsep-konsep untuk menerangkan data, hubungan-hubungan antara data dan

batasan-batasan data yang terintegrasi di dalam suatu organisasi. [13]

2.10 *Database Cluster*

Database clustering adalah kumpulan dari beberapa server yang berdiri sendiri yang kemudian bekerjasama sebagai suatu sistem tunggal. Saat ini aplikasi *database* semakin berkembang, baik dalam hal kegunaan, ukuran, maupun kompleksitas. Hal ini secara langsung berdampak pada server *database* sebagai penyedia layanan terhadap akses *database*, konsekuensi dari semua itu adalah beban *database* server akan semakin bertambah berat dan mengakibatkan kurang optimalnya kinerja dari *server* tersebut.

Oleh karena itu diperlukan perancangan yang tepat dan handal dalam membangun *database* server. *Database* pada masa sekarang ini dituntut agar dapat berjalan dengan cepat, mempunyai kehandalan dan keseterdiaan yang tinggi, dengan *clustering database* yang disimpan dapat terbagi ke beberapa mesin dan pada saat aplikasi berjalan, semua mesin yang menyimpan data tersebut dianggap sebagai satu kesatuan. Metode *clustering* seperti ini sangat baik untuk *loadbalancing* dan penanganan *system failure* karena kemampuan tiap mesin akan digunakan dan jika ada salah satu mesin yang mengalami *failure* maka sistem tidak akan langsung terganggu karena mesin lain akan tetap berfungsi. Kemampuan *clustering* memungkinkan sebuah *database* tetap hidup dalam waktu yang lama.

2.11 *MySQL Cluster*

MySQL Cluster merupakan sebuah tipe basis data (*database*) yang dapat beroperasi dalam ukuran data yang relatif besar (maksimal dalam skala beberapa ratus gigabyte). *MySQL Cluster* adalah sebuah teknologi baru untuk memungkinkan clustering di dalam *memory database* dalam sebuah sistem *share-nothing*. Arsitektur *share-nothing* mengijinkan sistem dapat bekerja dengan *hardware*/perangkat keras yang sangat murah, dan tidak membutuhkan perangkat keras dan lunak dengan spesifikasi

khusus. Arsitektur tersebut juga handal karena masing-masing komponen mempunyai memory dan disk tersendiri. *MySQL Cluster* menggabungkan *MySQL* Server biasa dengan sebuah mesin penyimpanan *in-memory* tercluster yang dinamakan NDB. NDB berarti bagian dari suatu rangkaian yang dikhususkan sebagai mesin penyimpanan, sedangkan *MySQL Cluster* diartikan sebagai kombinasi atau gabungan dari *MySQL* dan mesin penyimpanan yang baru tersebut[1].

2.12 Arsitektur *MySQL Cluster*

MySQL Cluster merupakan sebuah database yang menggunakan arsitektur shared-nothing dan antarmuka SQL yang telah umum digunakan. Sistem *database* ini terdiri dari beberapa node yang dapat didistribusikan ke beberapa perangkat keras dan ke beberapa wilayah/zona yang berbeda sekaligus untuk tetap menjaga ketersediaan data meskipun jaringan ataupun salah satu node sedang mengalami kegagalan (*failure*). Ada tiga node yang menyusun *MySQL Cluster*, yakni:

1. *Data Nodes*, digunakan untuk menyimpan semua data yang menjadi milik *MySQL Cluster*. Semua data direplikasi di node-node ini.
2. *Management Server Nodes*, digunakan untuk mengendalikan konfigurasi sistem ketika startup. Selain itu, node ini juga dapat digunakan sebagai pengidentifikasi setiap perubahan setting yang terjadi pada *cluster*.
3. *MySQL Server Nodes*, berfungsi sebagai pintu akses untuk masuk ke dalam node-node data yang ter-*cluster*.

2.13 PHP (*Hypertext Preprocessor*)

PHP adalah singkatan dari "PHP: *Hypertext Prepocessor*", yaitu bahasa pemrograman yang digunakan secara luas untuk penanganan pembuatan dan pengembangan sebuah situs web dan bisa digunakan bersamaan dengan HTML. PHP diciptakan oleh

---

[1] https://www.mysql.com/products/cluster/

Rasmus Lerdorf pertama kali tahun 1994. Pada awalnya PHP adalah singkatan dari "Personal Home Page Tools". Selanjutnya diganti menjadi FI ("*Forms Interpreter*"). Sejak versi 3.0, nama bahasa ini diubah menjadi "PHP: *Hypertext Prepocessor*" dengan singkatannya "PHP". PHP versi terbaru adalah versi ke-5. Berdasarkan survey Netcraft pada bulan Desember 1999, lebih dari sejuta site menggunakan PHP, di antaranya adalah NASA, Mitsubishi, dan RedHat.

2.14 *Framework Yii*

Yii 2.0 adalah sebuah kerangka kerja untuk bahasa pemrograman PHP dimana kerangka kerja ini telah didukung oleh aspek *modularity* dan *eloquent database*[2]. Arsitektur perangkat lunak ini menggunakan *Model-View-Controller*, namun dengan aspek *modularity*, maka dapat dibuat modul dimana setiap modul memiliki MVC tersendiri. Sedangkan *eloquent database* adalah basis data berupa objek di dalam sebuah bahasa pemrograman PHP.

[2] http://www.yiiframework.com/doc-2.0/guide-intro-yii.html

[Halaman ini segaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan Tugas Akhir. Analisis permasalahan membahas permasalahan yang yang diangkat dalam pengerjaan Tugas Akhir. Solusi yang ditawarkan oleh penulis juga dicantumkan pada tahap permasalahan analisis ini. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan perangkat lunak. Selanjutnya dibahas mengenai perancangan sistem yang dibuat. Perancangan direpresentasikan dengan diagram UML (*Unified Modelling Language*).

3.1. Analisis

Tahap analisis dibagi menjadi beberapa bagian antara lain cakupan permasalahan, deskripsi umum sistem, kasus penggunaan sistem, dan kebutuhan perangkat lunak.

3.1.1 Analisis Proses Bisnis

Aplikasi ERP bukan aplikasi yang baru muncul atau jarang digunakan. Namun aplikasi ERP merupakan aplikasi yang sudah lama ada. Namun masing-masing ERP memiliki proses bisnis yang berbeda-beda. Kebanyakan aplikasi ERP yang kompleks dan sesuai dengan banyak perusahaan memiliki harga jual yang tinggi. Permasalahan berada pada bagaimana business plan yang telah ada dapat dijalankan secara efisien dengan aplikasi ERP ini. Setiap ERP memiliki proses bisnis yang berbeda-beda. Aplikasi ERP pada Tugas Akhir ini memiliki proses bisnis sendiri dan dibandingan dengan aplikasi ERP yang cukup banyak digunakan banyak perusahaan. Seperti Odoo, Adempiere, dan InoERP. Pada bab ini akan dijelaskan tentang analisa proses bisnis yang telah ada.

Sesuai yang sudah dijelaskan pada subbab 2.8 dan juga menganalisa proses bisnis yang dimiliki oleh beberapa aplikasi serupa dimana proses bisnis dari aplikasi serupa dapat dilihat di

lampiran Gambar B.11, Gambar B.12 dan Gambar B.13. Berikut proses bisnis yang dimiliki oleh Odoo, Adempiere, dan InoERP jika dibandingkan dengan ERP yang dikerjakan :

Table 3.1 Kekurangan dan Kelebihan pada Odoo, Adempiere, dan InoERP

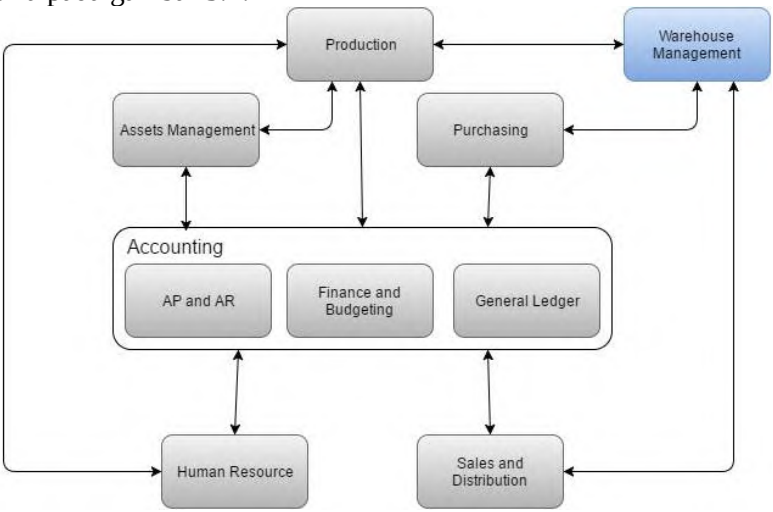
Nama Modul	Odoo	Adempiere	InoERP
Inventory	Kontra: Desain pada Inventory tidak terlalu mendetailkan perpindahan barang. Lebih mengutamakan proses pengelolaan barang yang sederhana.	Kontra: 1. Desain inventory pada InoERP: Warehouse->Locator->Storage Desain inventory pada EZERP: Warehouse->Inventory->Storage->SubStorage->Bin	Kontra: 1. Desain inventory pada InoERP: Inventory->SubInventory->Locator Desain inventory pada EZERP: Warehouse->Inventory->Storage->SubStorage->Bin 2. Pencatatan pengiriman barang diluar gudang tidak ada

Tabel 3.1 menunjukkan bahwa masih banyak kekurangan atau tidak memenuhi kebutuhan dari *business plan* yang telah dibuat. Pada modul HRM dapat dilihat bahwa modul HR pada Odoo, Adempiere dan InoERP belum ada submodul KPI dalam menentukan bonus. Selain itu untuk penggajian pegawai masih secara manual satu per satu.

Berdasarkan kekurangan yang dimiliki oleh setiap ERP maka perlu dikembangkannya sebuah aplikasi ERP yang sesuai dengan kebutuhan pada *business plan.* Terdapat dua jenis proses bisnis yang digunakan, yakni *Make-To-Order* (MTO) dan *Make-*

To-Stock (MTS). MTO adalah sebuah proses bisnis dimana produksi didasarkan pada permintaan. MTS adalah sebuah proses bisnis dimana produksi didasarkan oleh *sales-forecasting*. Berikut proses bisnis yang dirancang sesuai dengan kebutuhan *business plan* dimana proses bisnis ini telah mengakomodasi kekurangan Odoo, Adempiere, dan InoERP.

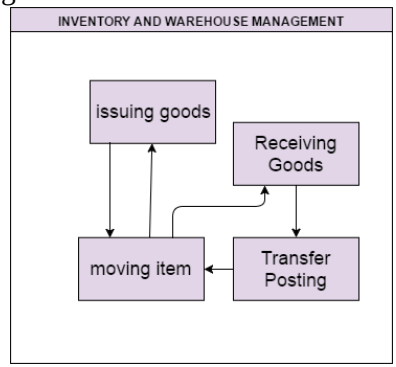
Berdasarkan proses bisnis yang terdapat pada ERP yang dijelaskan pada subbab 2.2.1. Berikut hasil rancangan proses bisnis yang dirangkum menjadi 9 modul dan memiliki hubungan keterkaitan yang dijelaskan pada gambar 3.1. proses bisnis ini menjelaskan hubungan antar modul untuk menjalankan proses bisnis yang sudah dijelaskan pada subbab 2.2.1. Modul yang akan dikerjakan ditunjukkan dengan modul yang diwarnai dengan warna biru pada gambar 3.1.



Gambar 3.1 Level 0 proses bisnis ERP

Proses bisnis utama yang akan dikerjakan pada sistem ini adalah *make-to-order* (MTO) dan *make-to-stock* (MTS) dimana 2 proses bisnis ini dapat dijelaskan lebih lanjut pada gambar 3.3 dan gambar 3.4 dimana 2 proses bisnis ini adalah proses bisnis level 1 turunan dari level 0 yang ditunjukkan pada gambar 3.1. Hubungan

antar modul yang ditunjukkan pada level 0 diturunkan ke level 1 dan dikelompokkan menjadi 2 proses bisnis yakni MTO dan MTS. Untuk lebih menjelaskan submodul utama pada modul IWM dijelaskan pada gambar 3.2.



Gambar 3.2 Level 1 Proses Bisnis Modul IWM ERP

Pada Tugas Akhir ini dibangun aplikasi *Inventory and Warehouse Management* yang terintegrasi ke dalam sebuah sistem *Enterprise Resource Planning* (ERP). Proses bisnis yang terdapat pada modul IWM dijelaskan pada gambar 3.2 dimana memiliki 4 modul traksaksi utama yakni :

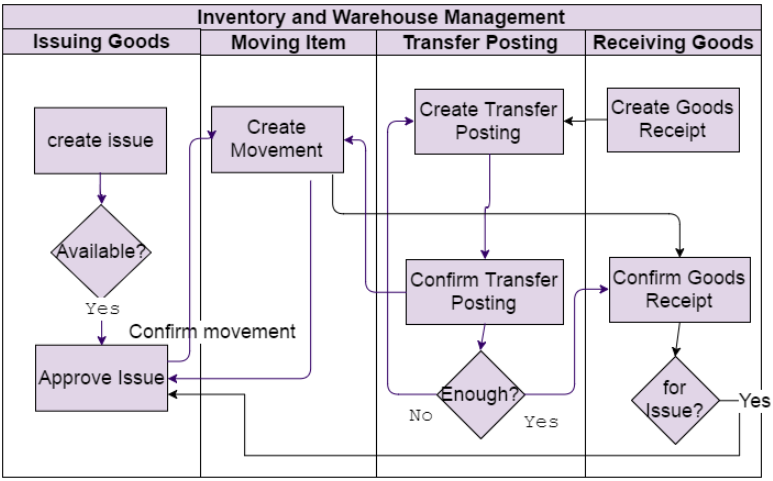
1. *Good Receipt* : Adalah Submodul yang bertugas untuk mencatat segala pemasukan barang ke gudang, baik melalui pembelian maupun hasil produksi.
2. *Good Issue* : Adalah submodul yang bertugas mencatat segala permintaan barang baik dari Penjualan, Produksi dan Aset.
3. *Transfer Posting* : Adalah submodul yang bertugas untuk mencatat segala perpindahan barang yang terjadi di luar gudang. Baik perpindahan barang antar gudang maupun pengiriman barang kepada pembeli.
4. *Warehouse Movement* : Adalah submodul yang bertugas untuk mencatat segala perpindahan barang yang terdapat pada sebuah gudang dimana bertugas sebagai *log* yang akan mencatat segala history perpindahan barang baik barang keluar gudang (*Good Issue*) maupun penerimaan barang masuk gudang (*Good Receipt*).

Untuk lebih menjelaskan kondisi barang yang terdapat pada gudang, berikut status dari barang yang akan digunakan dalam proses bisnis yang sudah dijelaskan :

1. *Unrestricted Use* : Adalah status dari barang dimana barang tersebut dalam keadaan kosong atau tanpa status. Status ini bisa diasumsikan sebagai stok barang dalam gudang yang masih belum digunakan atau dapat digunakan jika terdapat permintaan terhadap barang tersebut.
2. *Reserved* : Adalah status dari barang dimana barang tersebut dalam kondisi masih berada di dalam gudang namun memiliki status "*Reserve*" dimana barang tersebut tidak boleh digunakan lagi untuk keperluan lain. Barang tersebut hanya bisa digunakan oleh divisi yang memesan barang tersebut.
3. *In Quality Inspection* : Adalah status dari barang dimana barang tersebut berada dalam tahap pemindahan dari lokasi barang tersebut disimpan ke sektor pengiriman untuk diproses lebih lanjut sebelum barang tersebut dikirimkan kepada pemesan dari barang tersebut keluar gudang.
4. *Delivering* : Adalah status dari barang dimana barang tersebut secara fisik sudah dalam proses pengiriman dan sudah tidak berada pada gudang lagi namun pada sistem masih tersimpan untuk proses lebih lanjut.
5. *Used* : Adalah status dari barang dimana barang tersebut secara fisik sudah berubah bentuk menjadi barang hasil produksi namun secara sistem masih tersimpan untuk proses penentuan harga dari barang hasil produksi.

Tujuan dari aplikasi *Inventory and Warehouse Management* adalah untuk mengontrol setiap pergerakan barang yang terdapat pada sebuah perusahaan. Sistem yang dibangun berorientasi *multi-tenancy* dengan basis data terdisitribusi.

Proses bisnis yang terdapat pada IWM ERP dapat dipecah kembali lebih detail yang dapat dilihat pada gambar 3.3.



Gambar 3.3 Level 2 Proses Bisnis IWM ERP

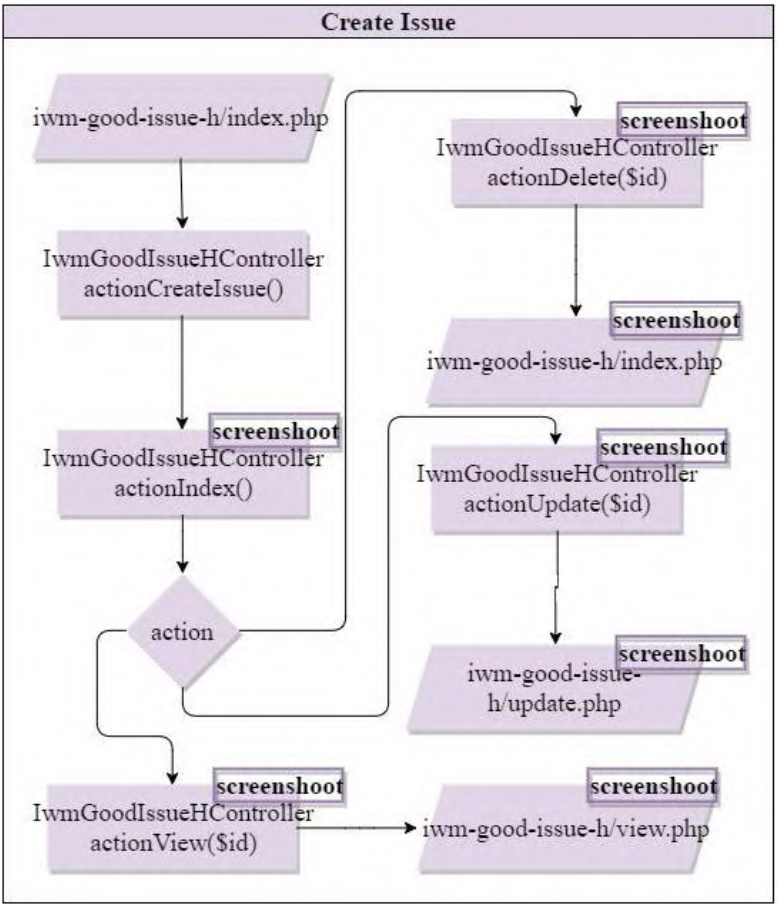
Untuk menjelaskan proses-proses yang terdapat pada setiap submodul IWM ERP, dapat dilihat pada Tabel 3.2.

Tabel 3.2 Proses Bisnis Level 2

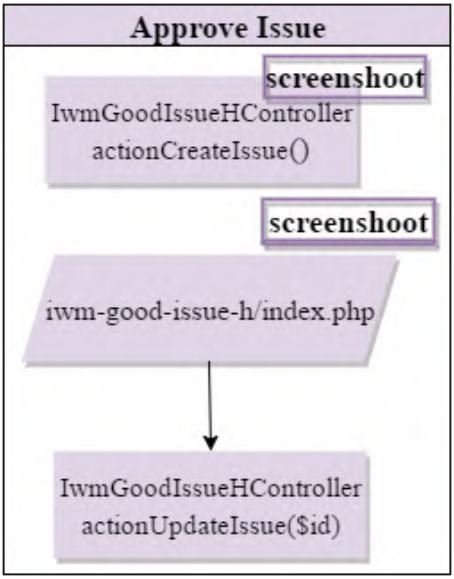
NO	PROSES BISNIS	KETERANGAN
1	Create Issue	Proses ini menyiapkan data *issue* yang dilakukan oleh pihak kedua, yakni divisi produksi, penjualan dan asset
2	Approve Issue	Proses ini yakni memproses lebih lanjut mengenai permintaan barang (*issue*) yang diminta oleh pihak kedua. Proses ini baru akan berjalan ketika semua barang yang diminta sudah siap dan sesuai dengan jumlah permintaan.
3	Create Movement	Proses ini membuat catatan dan mengatur perpindahan barang sesuai dengan konsep FIFO
4	Create Transfer Posting	Proses ini membuat catatan pengiriman barang di luar barang, proses ini mencatat ketika terdapat pengiriman baik ke dalam

NO	PROSES BISNIS	KETERANGAN
		(barang datang) maupun keluar gudang (pengiriman barang)
5	Confirm Transfer Posting	Proses ini melakukan konfirmasi terhadap proses pengiriman barang dimana baru dilakukan ketika proses pengiriman baik barang datang atau pengiriman barang sudah selesai.
6	Create Goods Receipt	Proses ini membuat sebuah catatan *receipt* dimana berisi catatan barang yang datang baik dari pembelian maupun hasil produksi berikut dengan jumlah barang dan catatan barang rusak jika ada.
7	Confirm Goods Receipt	Proses ini mengkonfirmasi proses *receipt* jika seluruh barang sudah tiba dan tercatat dalam system.

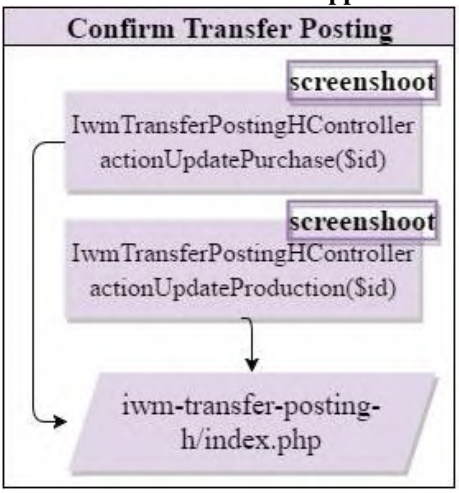
Proses bisnis yang terdapat pada setiap submodul yang terdapat pada modul ERP yang ditunjukkan oleh gambar 3.3 dapat dipecah dan dijelaskan dengan lebih terperinci dimana setiap bisnis proses dijelaskan dengan sebuah gambar. Untuk setiap proses bisnis dapat dilihat pada gambar 3.4, gambar 3.5, gambar 3.6, gambar 3.7, gambar 3.8, gambar 3.9, dan gambar 3.10. Pada setiap gambar yang disebutkan, menjelaskan proses bisnis secara sistem dimana menjelaskan alur data sesuai dengan fungsi yang dibutuhkan yakni *Create, Read, Update, Delete*.



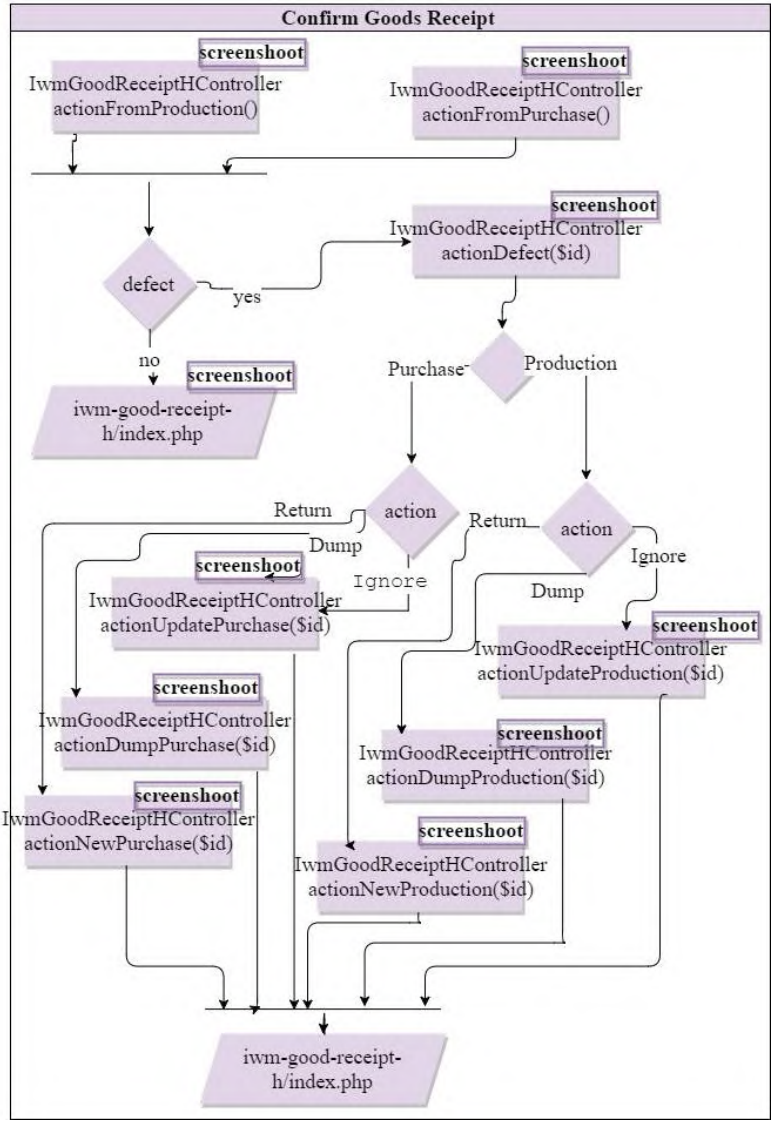
Gambar 3.4 Level 3 Proses Create Issue IWM ERP



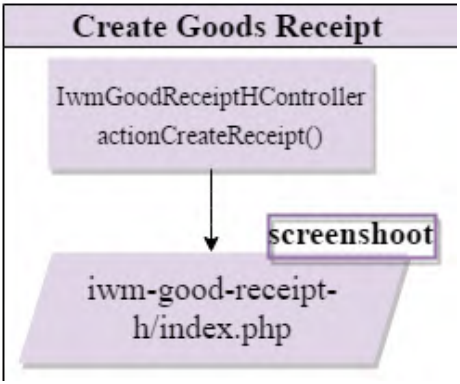
Gambar 3.5 Level 3 Proses Approve Issue



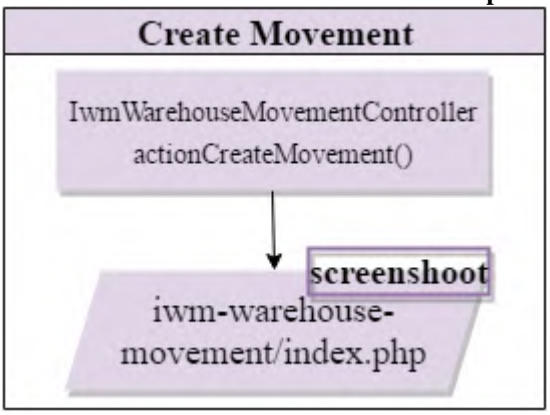
Gambar 3.6 Level 3 Confirm Transfer Posting



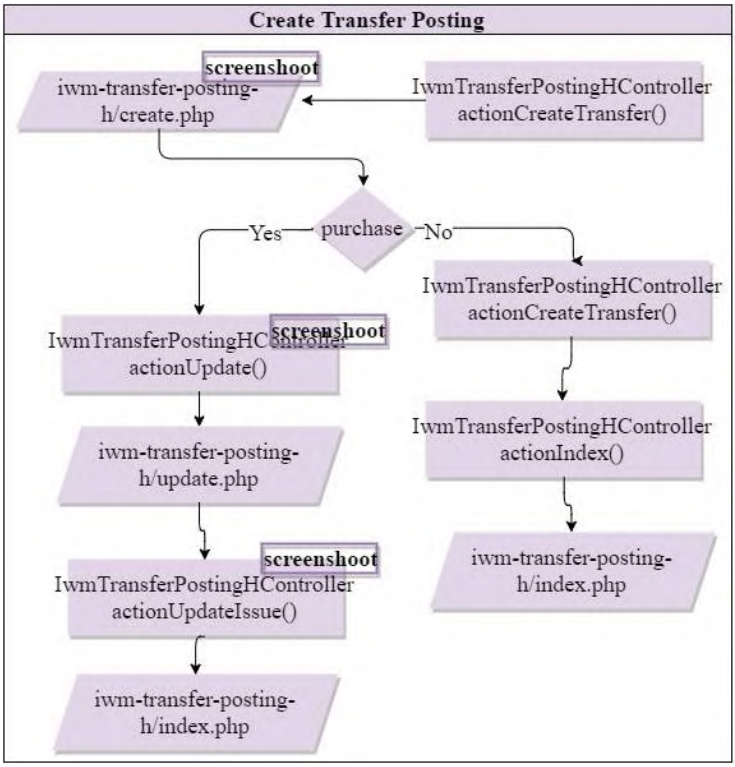
Gambar 3.7 Level 3 Confirm Good Receipt



Gambar 3.8 Level 3 Create Good Receipt



Gambar 3.9 Level 3 Create Movement



Gambar 3.10 Level 3 Create Transfer Posting

3.1.2 **Analisis Data**

Konsep autentikasi dan otorisasi data yang digunakan adalah *multitenancy separated database, separated schema*. Sesuai yang sudah dijelaskan pada subbab 2.5.1 mengenai arsitektur *multitenancy* dimana arsitektur *separated database, separated schema* dimana pada konsep ini, data milik setiap *tenant* akan disimpan di basis data yang terpisah. Sehingga tidak ada 2 atau lebih *tenant* menggunakan tabel dan basis data yang sama. Penambahan *tenant* baru akan menyebabkan penambahan sebuah basis data baru. Penggunaan konsep *separated database, separated schema* dimaksudkan agar data milik setiap *tenant* dapat terjamin

kerahasiaannya. Keuntungan lainnya adalah kerumitan proses *maintenance* basis data dapat dikurangi.

Konsep basis data terdistribusi yang diterapkan adalah replikasi. Sesuai dengan yang sudah dijelaskan pada subbab 2.7.1, replikasi memungkinkan penyalinan setiap tabel basis data ke *node-node* penyusun basis data terdistribusi. Kegagalan sebuah *node* tidak akan menyebabkan basis data berhenti bekerja. Sebaliknya, hal tersebut akan memicu mekanisme sinkronisasi jika *node* yang mati kembali hidup.

3.2 Deskripsi Umum Sistem

Tahap ini meliputi perancangan basis data, tampilan antarmuka, dan perancangan alur proses penggunaan system pada modul IWM yang diharapkan dapat memenuhi tujuan dari pengembangan aplikasi ini.

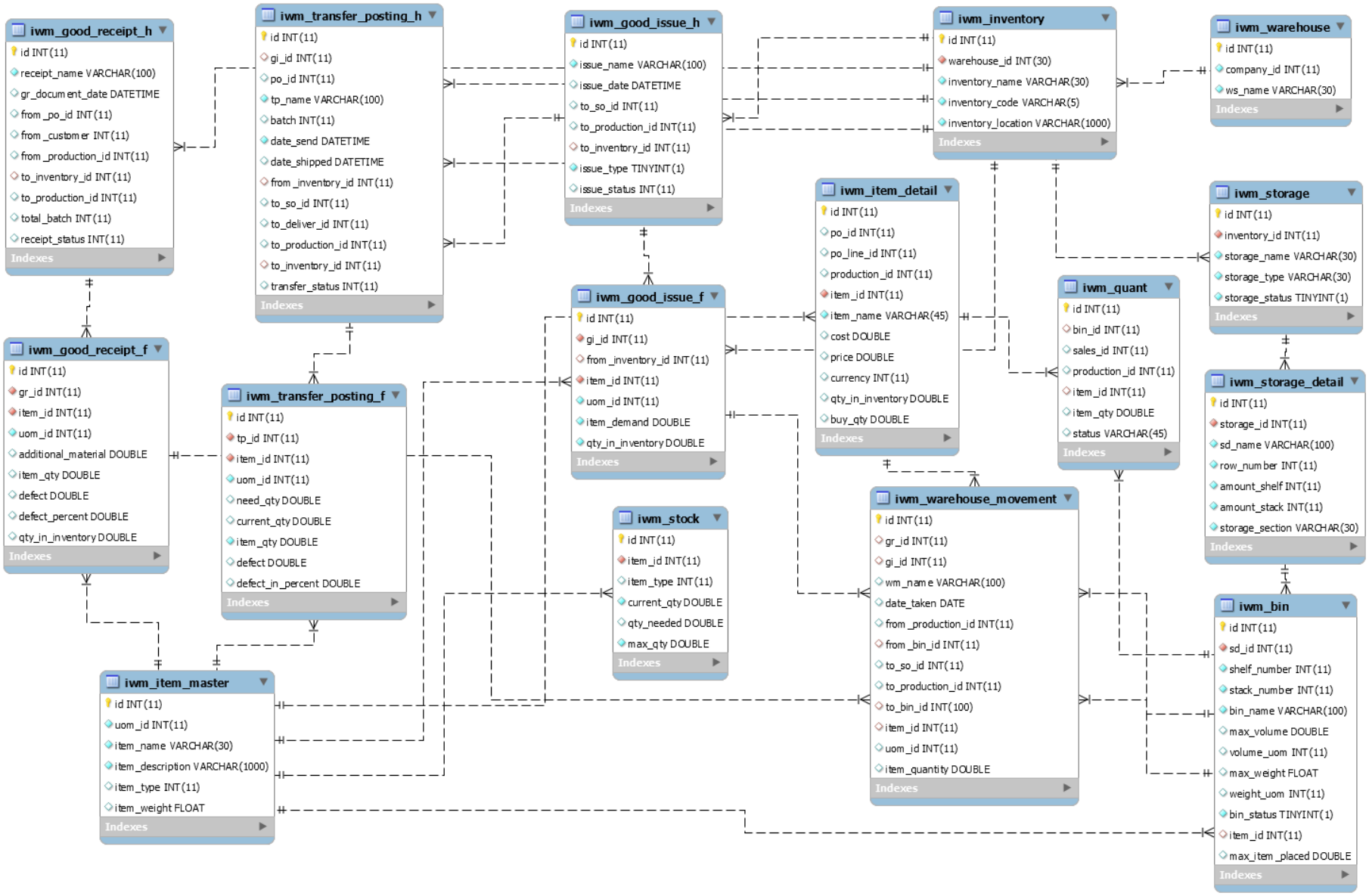
3.2.1 Identifikasi Pengguna

Pada sistem yang akan dibangun, aktor yang akan menjadi pengguna sistem adalah *staff* gudang dan manager gudang. *Staff* gudang mempunyai hak akses dalam pengelolaan data barang dan juga pengelolaan gudang, yang terdiri atas membuat (*create*), memperbarui (*update*) dan menghapus (*delete*). Sedangkan manager mempunyai hak akses melihat (*view*) data yang telah dikelola oleh *staff*.

3.2.2 Perancangan Basis Data

Berdasarkan subbab 2.8 dan 2.9. dibutuhkan sebuah data model yang mampu mengimplementasikan proses bisnis yang didapat dalam memproses bisnis proses MTO dan MTS pada Invetory. Berdasarkan hal tersebut, didapatkan PDM dari bisnis proses IWM dimana dijelaskan pada gambar 3.3.

[Halaman ini sengaja dikosongkan]

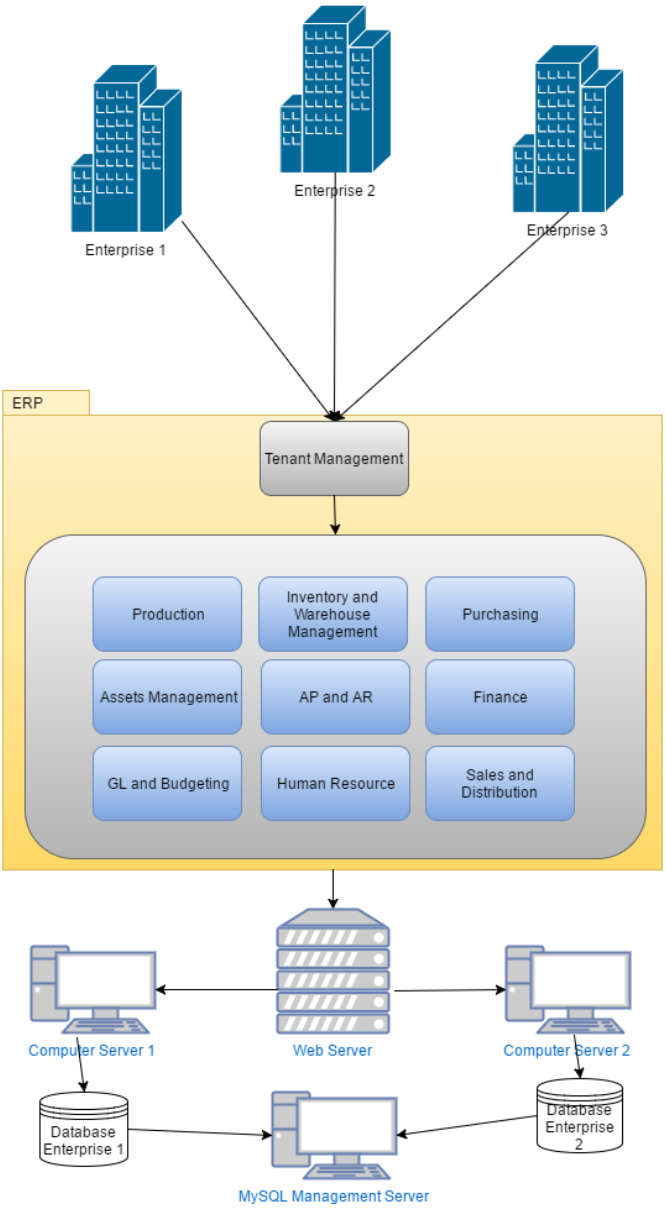


Gambar 3.11 PDM IWM ERP

[Halaman ini sengaja dikosongkan]

3.2.3 **Arsitektur Sistem**

Dalam mengimplementasikan *Multitenancy* dengan basis data terdistribusi. Dikembangkan sistem ERP ini menggunakan sebuah *tenant management* sesuai dengan subbab 2.5 dimana *tenant management* bertugas dalam mengarahkan *tenant* dalam menggunakan *database* pada perusahaan *tenant* tersebut. dan 4 buah server untuk basis data terdistribusi sesuai dengan subbab 2.7 dimana terdiri dari sebuah *web server* yang bertugas menyimpan aplikasi ERP tersebut, 2 buah *database server* dimana bertugas menyimpan seluruh transaksi yang terdapat pada sistem ERP untuk seluruh *tenant* dan sebuah *management server* yang bertugas untuk melakukan replikasi pada kedua buah *database server* tersebut. Untuk lebih jelasnya dapat dilihat pada gambar 3.4.



Gambar 3.12 Diagram Basis Data Terdistribusi

3.2.4 Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan dalam sistem ini mencakup kebutuhan fungsional. Kebutuhan fungsional berisikan proses-proses yang dibutuhkan dalam sistem dan harus dijalankan. Kebutuhan fungsional sistem dideskripsikan dalam Tabel 3.2

Tabel 3.3 Daftar Kebutuhan Fungsional Sistem

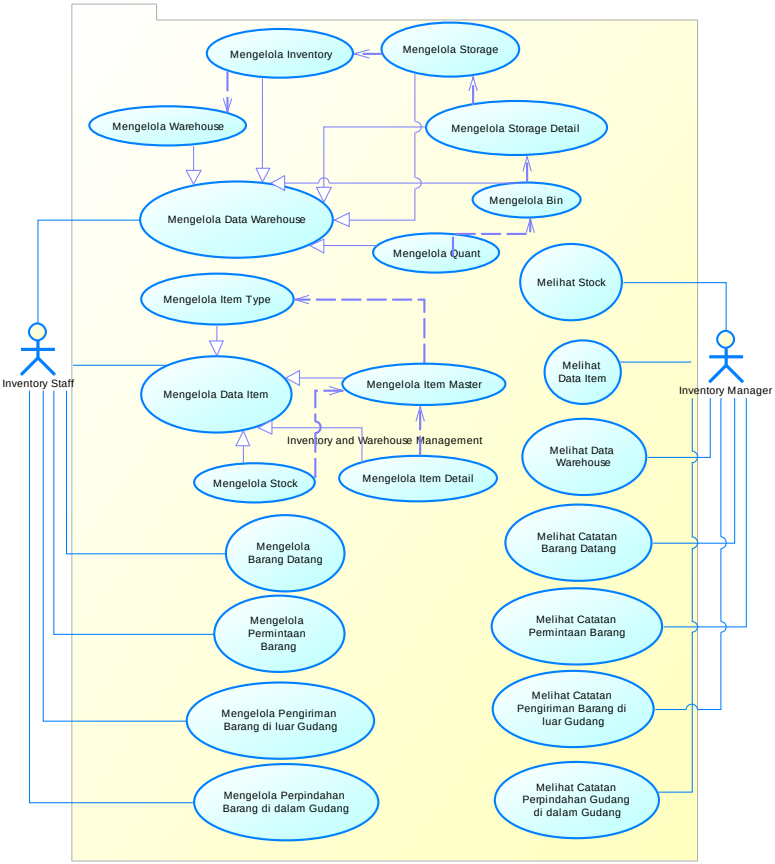
Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-001	Mengelola Item Master	Pengguna dapat mengelola data item master yang terbagi ke dalam beberapa bagian, yaitu: Item Type, Item Master, Item Detail, Stock, dan Quant serta pada setiap bagian dapat mengelola data tersebut meliputi proses: *create, update, delete*
F-002	Mengelola Warehouse Master	Pengguna dapat mengelola data warehouse yang terbagi ke dalam beberapa bagian, yaitu: Warehouse, Inventory, Storage, Storage Detail, dan Bin serta pada setiap bagian dapat mengelola data tersebut meliputi proses: *create, update, delete*
F-003	Mengelola Barang Datang	Pengguna dapat mengelola pemasukan barang yang meliputi

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
		proses: *create, update, delete*
F-004	Mengelola Permintaan Barang	Pengguna dapat mengelola permintaan barang yang meliputi proses: *create, update, delete*
F-005	Mengelola Perpindahan dalam Gudang	Pengguna dapat mengelola perpindahan barang dalam gudang yang meliputi proses: *create, update, delete*
F-006	Mengelola Perpindahan di luar Gudang	Pengguna dapat mengelola perpindahan di luar gudang yang meliputi proses: *create, update, delete*
F-007	Melihat Stok Barang di Gudang	Pengguna dapat melihat jumlah stok yang terdapat di gudang pada saat itu
F-008	Melihat Data Item Master	Pengguna dapat melihat data item master
F-009	Melihat Data Warehouse Master	Pengguna dapat melihat data warehouse master
F-010	Melihat Data Barang Datang	Pengguna dapat melihat data barang dating
F-011	Melihat Data Permintaan Barang	Pengguna dapat melihat data permintaan barang
F-012	Melihat Data Pengiriman Barang di Luar Gudang	Pengguna dapat melihat data pengiriman barang di luar gudang

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
F-013	Melihat Data Perpindahan Barang di Dalam Gudang	Pengguna dapat melihat data perpindahan barang di dalam gudang
F-014	Melihat Report Jumlah Perpindahan Barang	Pengguna dapat melihat report jumlah perpindahan barang setiap harinya baik itu barang datang maupun barang keluar gudang

3.2.5 **Kasus Penggunaan**

Kasus penggunaan yang dibutuhkan pada sistem sesuai dengan analisa yang telah dilakukan. Diagram kasus penggunaan dapat dilihat pada Gambar 3.1.5.1 dan kode kasus penggunaan ada pada tabel 3.2.



Gambar 3.3.2.5.1 Diagram Kasus Penggunaan

Tabel 3.4 Keterangan Kode Kasus Penggunaan

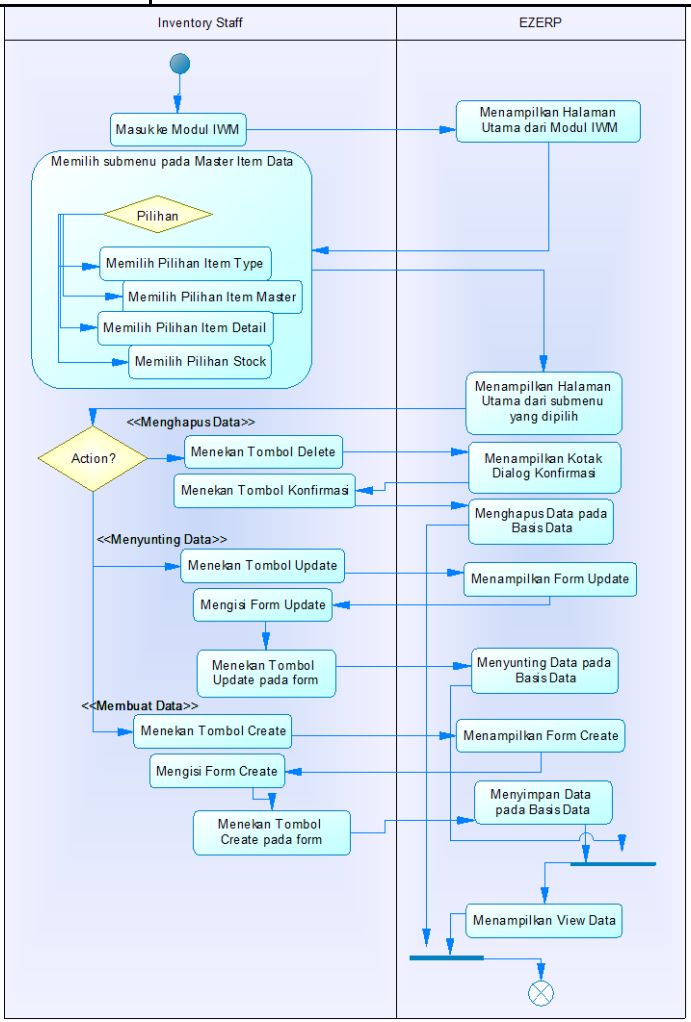
Kode Kasus Penggunaan	Kasus Penggunaan
UC-001	Mengelola Item Master
UC-002	Mengelola Warehouse Master
UC-003	Mengelola Barang Datang
UC-004	Mengelola Permintaan Barang
UC-005	Mengelola Perpindahan dalam Gudang
UC-006	Mengelola Perpindahan di luar Gudang
UC-007	Melihat Stok Barang di Gudang
UC-008	Melihat Data Item Master
UC-009	Melihat Data Warehouse Master
UC-010	Melihat Data Barang Datang
UC-011	Melihat Data Permintaan Barang
UC-012	Melihat Data Pengiriman Barang di Luar Gudang
UC-013	Melihat Data Perpindahan Barang di Dalam Gudang
UC-014	Melihat Report Jumlah Perpindahan Barang
UC-RBAC	Melakukan proses Login dan mengelola User

3.2.5.1 UC-001 Mengelola Item Master

Tabel 3.5 Spesifikasi Kasus Penggunaan Mengelola Item Master

Nama	Mengelola Item Master
Nomor	UC-001
Description	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data Item Master yang terdiri dari submodul (*Item Type, Item Master, Item Detail, Stock*)
Tipe	Fungsional
Aktor	Staf Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan salah satu submodul dari Master Item Data
Alur Normal	1. Staff membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Staff memilih salah satu submenu yang terdapat pada menu Item Master Data a. Submenu Item Type b. Submenu Item Master c. Submenu Item Detail d. Submenu Stock 4. Sistem menampilkan halaman utama pada submenu yang dipilih 5. Staff akan memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu 6. Kasus Penggunaan Berakhir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke alur normal langkah 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tersebut 2. Pengguna menyunting data tersebut 3. Sistem memperbaharui data yang sudah disunting 4. Berlanjut ke alur normal langkah 3

	A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke alur normal langkah 3



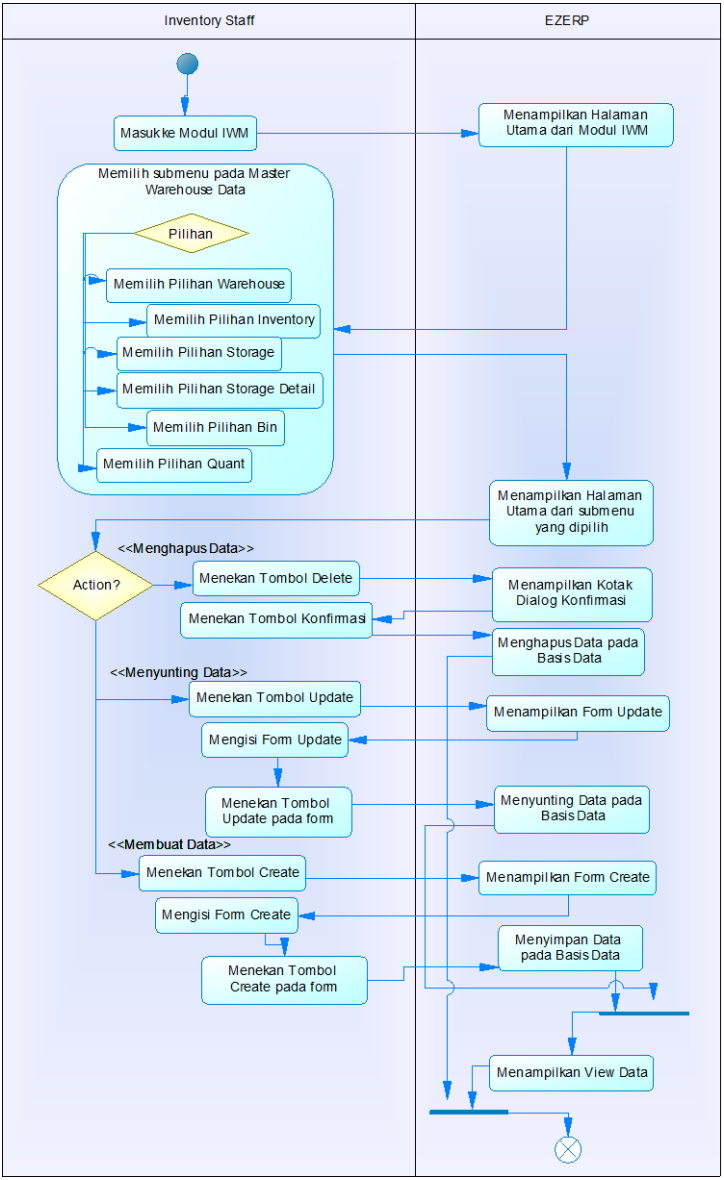
Gambar 3.3.13 Diagram aktivitas use case UC-001

3.2.5.2 UC-002 Mengelola Warehouse Master

Tabel 3.6 Spesifikasi Kasus Penggunaan Mengelola Warehouse Master

Nama	Mengelola Warehouse Master
Nomor	UC-002
Description	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data *warehouse master*
Tipe	Fungsional
Aktor	Staf Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan daftar salah satu submenu pada *Master Warehouse Data*
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih salah satu sub menu yang terdapat pada menu Warehouse Master Data a. Submenu Warehouse b. Submenu Inventory c. Submenu Storage d. Submenu Storage Detail e. Submenu Bin f. Submenu Quant 4. Sistem menampilkan halaman utama pada submenu yang dipilih 5. Aktor memilih kegiatan yang dapat dilakukan A1. Pengguna memilih membuat data baru A2. Pengguna memilih menyunting data tertentu A3. Pengguna memilih menghapus data tertentu 6. Kasus penggunaan berakhir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke alur normal langkah 3 A2. Pengguna memilih menyunting data tertentu

	1. Sistem menampilkan isian sunting data tersebut 2. Pengguna menyunting data tersebut 3. Sistem memperbaharui data yang sudah disunting 4. Berlanjut ke alur normal langkah 3 A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke alur normal langkah 3



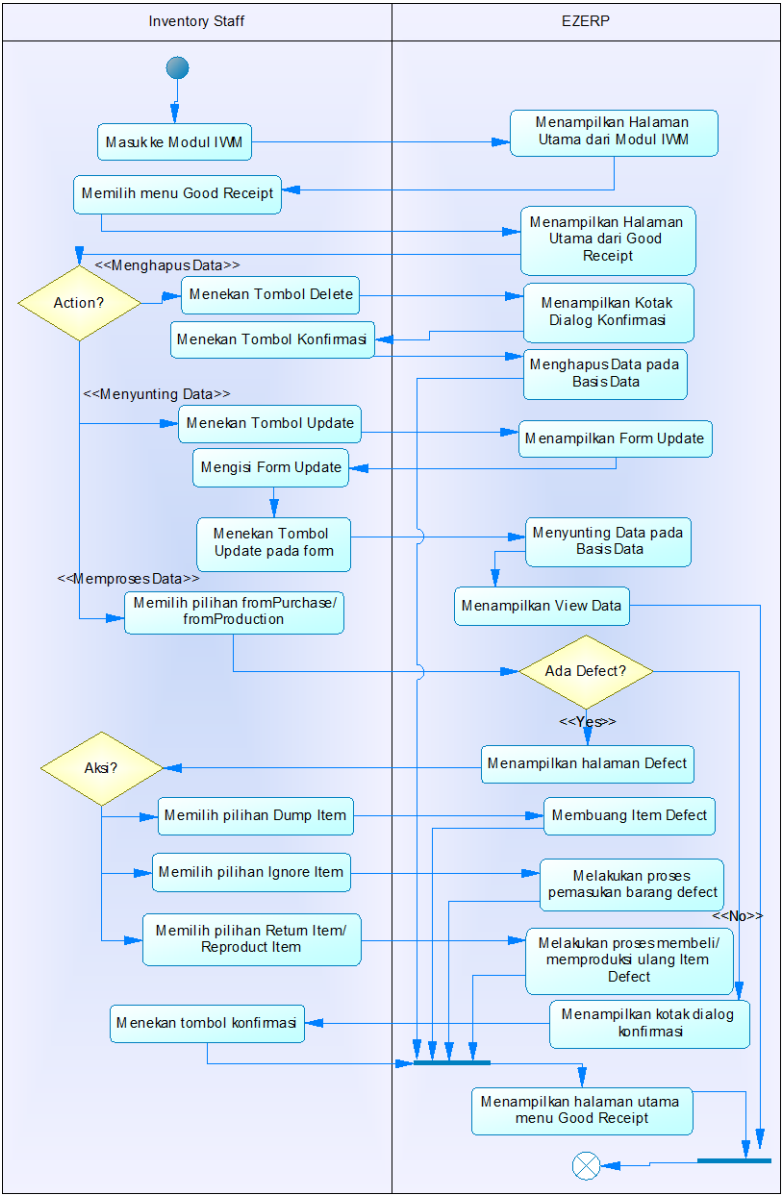
Gambar 3.3.14 Diagram aktivitas use case UC-002

3.2.5.3 UC-003 Mengelola Barang Datang

Tabel 3.7 Spesifikasi Kasus Penggunaan Mengelola Barang Datang

Nama	Mengelola Barang Datang
Nomor	UC-003
Description	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, memproses serta menghapus data penerimaan barang
Tipe	Fungsional
Aktor	Staf Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan daftar menu *Good Receipt*
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih menu Good Receipt 4. Sistem menampilkan halaman utama pada Good Receipt 5. Aktor memilih kegiatan yang dapat dilakukan A1. Aktor memilih menyunting data tertentu A2. Aktor memilih menghapus data tertentu A3. Aktor memilih memproses data tertentu 6. Kasus penggunaan berakhir
Alur Alternatif	A1. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tersebut 2. Pengguna menyunting data tersebut 3. Sistem memperbaharui data yang sudah disunting 4. Berlanjut ke alur normal langkah 3 A2. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke alur normal langkah 3 A3. Aktor memilih memproses data tertentu 1. Sistem akan melakukan salah satu kegiatan

	a. Jika pada data tertentu terdapat barang rusak, sistem akan menampilkan halaman tampilan barang rusak a1. Aktor akan memilih salah satu pilihan i. Aktor memilih Dump Item ii. Aktor memilih Ignore Item iii. Aktor memilih Return/Reproduct Item iv. Aktor memilih Cancel a2. Sistem akan memproses data tertentu sesuai dengan pilihan Aktor b. Jika pada data tertentu tidak terdapat barang rusak, sistem menampilkan pemberitahuan konfirmasi penyelesaian receipt b1. Aktor menyetujui penyelesaian proses data b2. Sistem akan memproses penyelesaian data tertentu 2. Berlanjut ke alur normal langkah 3



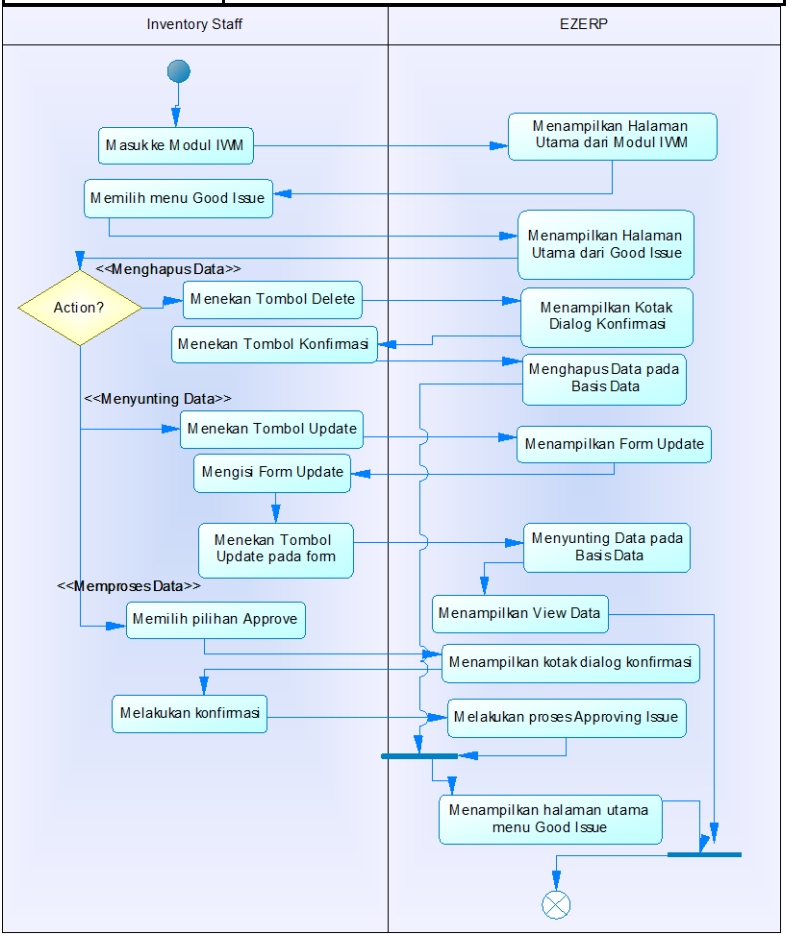
Gambar 3.3.15 Diagram aktivitas use case UC-003

3.2.5.4 UC-004 Mengelola Permintaan Barang

Tabel 3.8 Spesifikasi Kasus Penggunaan Mengelola Permintaan Barang

Nama	Mengelola Permintaan Barang
Nomor	UC-004
Description	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, memproses serta menghapus data permintaan barang
Tipe	Fungsional
Aktor	Staf Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan daftar menu *Good Issue*
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih menu Good Issue 4. Sistem menampilkan halaman utama pada Good Issue 5. Aktor memilih kegiatan yang dapat dilakukan A1. Aktor memilih menyunting data tertentu A2. Aktor memilih menghapus data tertentu A3. Aktor memilih memproses data tertentu 6. Kasus penggunaan berakhir
Alur Alternatif	A1. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tersebut 2. Pengguna menyunting data tersebut 3. Sistem memperbaharui data yang sudah disunting 4. Berlanjut ke alur normal langkah 3 A2. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke alur normal langkah 3 A3. Aktor memilih memproses data tertentu 1. Sistem menampilkan pemberitahuan konfirmasi penyelesaian data tertentu

	2. Aktor menyetujui penyelesaian proses data 3. Sistem akan memproses penyelesaian data tertentu 4. Berlanjut ke alur normal langkah 3



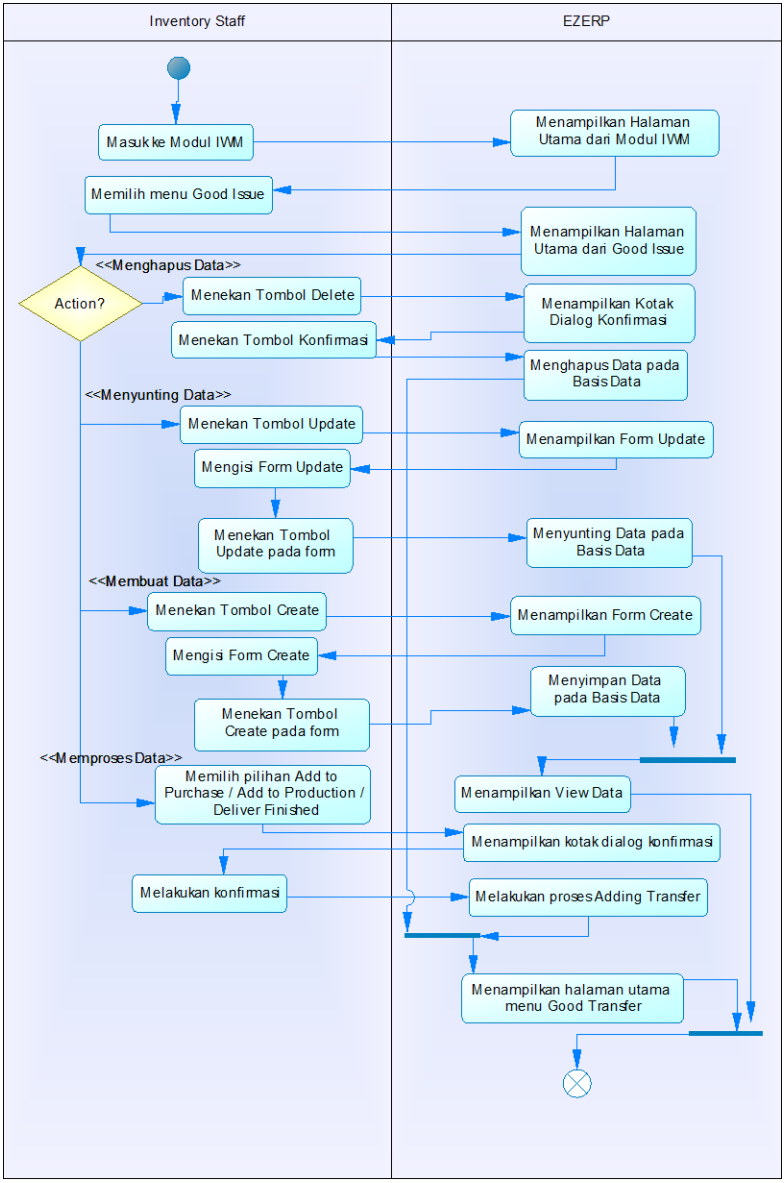
Gambar 3.3.16 Diagram aktivitas use case UC-004

3.2.5.5 UC-005 Mengelola Pengiriman Barang di Luar Gudang

Tabel 3.9 Spesifikasi Kasus Penggunaan Mengelola Pengiriman Barang di Luar Gudang

Nama	Mengelola Pengiriman Barang di Luar Gudang
Nomor	UC-005
Description	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, memproses serta menghapus data pengiriman barang baik pengiriman dari luar ke gudang maupun dari gudang keluar
Tipe	Fungsional
Aktor	Staf Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan daftar menu *Good Transfer*
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih menu Good Transfer 4. Sistem menampilkan halaman utama pada Good Transfer 5. Aktor memilih kegiatan yang dapat dilakukan A1. Aktor memilih membuat data baru A2. Aktor memilih menyunting data tertentu A3. Aktor memilih menghapus data tertentu A4. Aktor memilih memproses data tertentu 6. Kasus penggunaan berakhir
Alur Alternatif	A1. Pengguna memilih membuat data baru 1. Sistem menampilkan isian pembuatan data baru 2. Pengguna memasukkan data baru 3. Sistem menyimpan data baru 4. Berlanjut ke alur normal langkah 3 A2. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tersebut 2. Pengguna menyunting data tersebut 3. Sistem memperbaharui data yang sudah disunting 4. Berlanjut ke alur normal langkah 3

	A3. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke alur normal langkah 3 A4. Aktor memilih memproses data tertentu 1. Sistem menampilkan pemberitahuan konfirmasi penyelesaian data tertentu 2. Aktor menyetujui penyelesaian proses data 3. Sistem akan memproses penyelesaian data tertentu 4. Berlanjut ke alur normal langkah 3

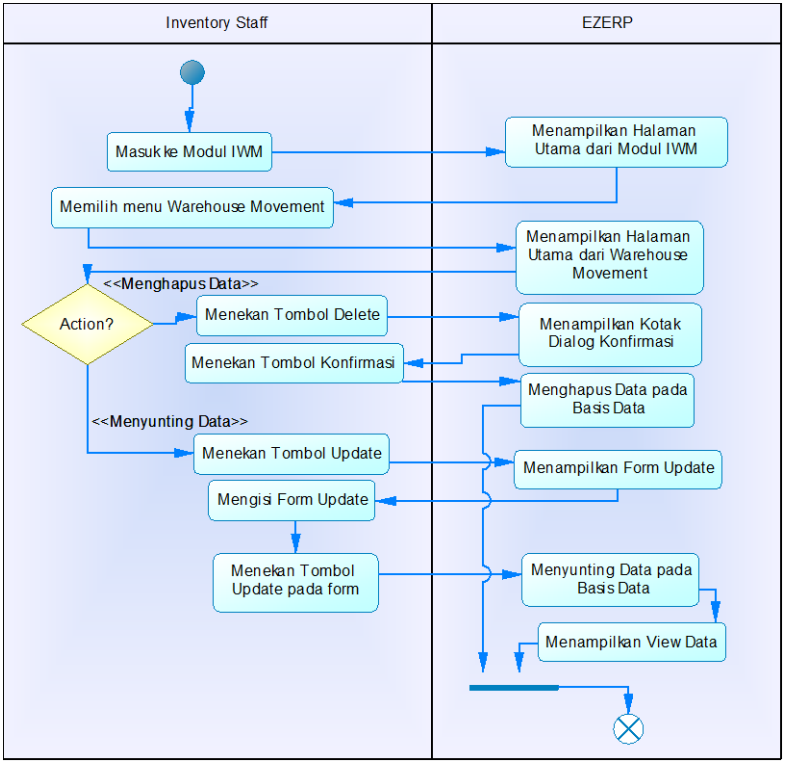


Gambar 3.3.17 Diagram aktivitas use case UC-005

3.2.5.6 UC-006 Mengelola Perpindahan Barang di Dalam Gudang

Tabel 3.10 Spesifikasi Kasus Penggunaan Mengelola Perpindahan Barang di Dalam Gudang

Nama	Mengelola Perpindahan Barang di Dalam Gudang
Nomor	UC-005
Description	Kasus penggunaan ini digunakan untuk melihat, menambah, menyunting, serta menghapus data perpindahan barang baik perpindahan dari luar ke dalam gudang maupun dari dalam gudang keluar
Tipe	Fungsional
Aktor	Staf Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan daftar menu *Warehouse Movement*
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih menu Warehouse Movement 4. Sistem menampilkan halaman utama pada Warehouse Movement 5. Aktor memilih kegiatan yang dapat dilakukan A1. Aktor memilih menyunting data tertentu A2. Aktor memilih menghapus data tertentu 6. Kasus penggunaan berakhir
Alur Alternatif	A1. Pengguna memilih menyunting data tertentu 1. Sistem menampilkan isian sunting data tersebut 2. Pengguna menyunting data tersebut 3. Sistem memperbaharui data yang sudah disunting 4. Berlanjut ke alur normal langkah 3 A2. Pengguna memilih menghapus data tertentu 1. Sistem menampilkan pemberitahuan penghapusan data tertentu 2. Pengguna menyetujui penghapusan data 3. Sistem menghapus data tersebut 4. Berlanjut ke alur normal langkah 3

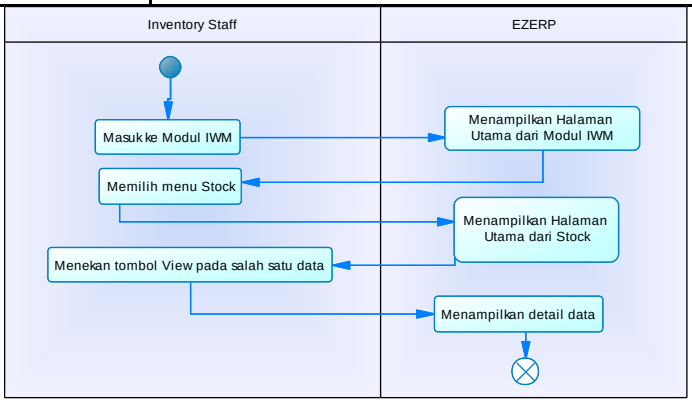


Gambar 3.3.18 Diagram aktivitas use case UC-006

3.2.5.7 UC-007 Melihat Stok di Gudang

Tabel 3.11 Spesifikasi Kasus Penggunaan Melihat Stok Barang di Gudang

Nama	Melihat Stok Barang di Gudang
Nomor	UC-007
Deskripsi	Kasus penggunaan ini digunakan untuk melihat jumlah stok seluruh barang di gudang
Tipe	Fungsional
Aktor	Staff Gudang, Manajer Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan daftar data stok di gudang
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih pilihan Stock 4. Sistem menampilkan halaman utama menu Stock 5. Aktor memilih pilihan melihat pada data tertentu 6. Sistem menampilkan detail data tertentu 7. Kasus penggunaan berakhir
Alur Alternatif	-

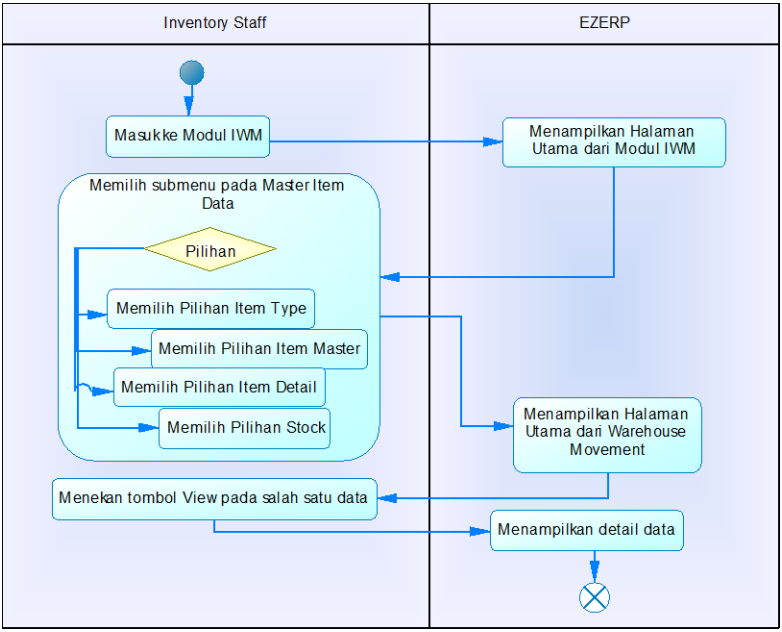


Gambar 3.3.19 Diagram aktivitas use case UC-007

3.2.5.8 UC-008 Melihat Data Item Master

Tabel 3.12 Spesifikasi Kasus Penggunaan Melihat Data Item Master

Nama	Melihat Data Item Master
Nomor	UC-008
Deskripsi	Kasus penggunaan ini digunakan untuk melihat seluruh data yang dikelola dalam Item Master
Tipe	Fungsional
Aktor	Manajer Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan salah satu data dari submenu Master Item Data
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih salah satu pilihan pada submenu Item Master Data a. Aktor memilih submenu Item Type b. Aktor memilih submenu Item Master c. Aktor memilih submenu Item Detail d. Aktor memilih submenu Stock 4. Sistem menampilkan halaman utama submenu pilihan 5. Aktor memilih pilihan melihat pada data tertentu 6. Sistem menampilkan detail data tertentu 7. Kasus penggunaan berakhir
Alur Alternatif	-



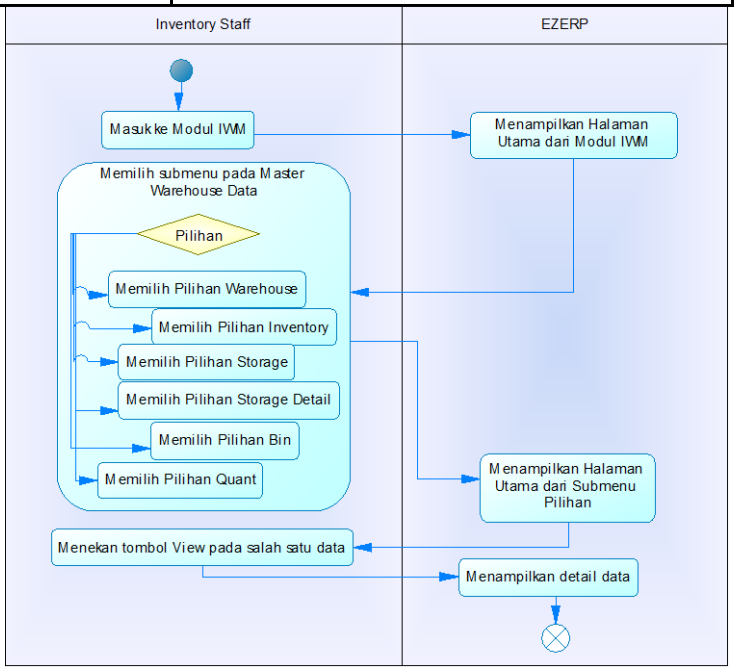
Gambar 3.3.20 Diagram aktivitas use case UC-008

3.2.5.9 UC-009 Melihat Data Warehouse Master

Tabel 3.13 Spesifikasi Kasus Penggunaan Melihat Data Warehouse Master

Nama	Melihat Data Warehouse Master
Nomor	UC-009
Deskripsi	Kasus penggunaan ini digunakan untuk melihat seluruh data yang dikelola dalam Warehouse Master
Tipe	Fungsional
Aktor	Manajer Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan salah satu data dari submenu Master Warehouse Data
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP

	2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih salah satu pilihan pada submenu Warehouse Master Data a. Aktor memilih submenu Warehouse b. Aktor memilih submenu Inventory c. Aktor memilih submenu Storage d. Aktor memilih submenu Storage Detail e. Aktor memilih submenu Bin f. Aktor memilih submenu Quant 4. Sistem menampilkan halaman utama submenu pilihan 5. Aktor memilih pilihan melihat pada data tertentu 6. Sistem menampilkan detail data tertentu 7. Kasus penggunaan berakhir
Alur Alternatif	-

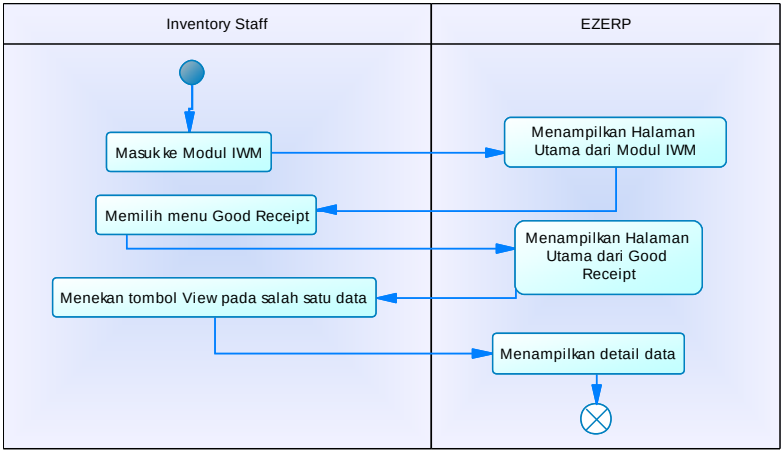


Gambar 3.3.21 Diagram aktivitas use case UC-009

3.2.5.10 UC-010 Melihat Data Barang Datang

Tabel 3.14 Spesifikasi Kasus Penggunaan Melihat Data Barang Datang

Nama	Melihat Data Barang Datang
Nomor	UC-010
Deskripsi	Kasus penggunaan ini digunakan untuk melihat seluruh data barang datang baik dari pembelian maupun hasil produksi
Tipe	Fungsional
Aktor	Manajer Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman utaman dari *Good Receipt*
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih menu Good Receipt 4. Sistem menampilkan halaman utama Good Receipt 5. Aktor memilih pilihan melihat pada data tertentu 6. Sistem menampilkan detail data tertentu 7. Kasus penggunaan berakhir
Alur Alternatif	-



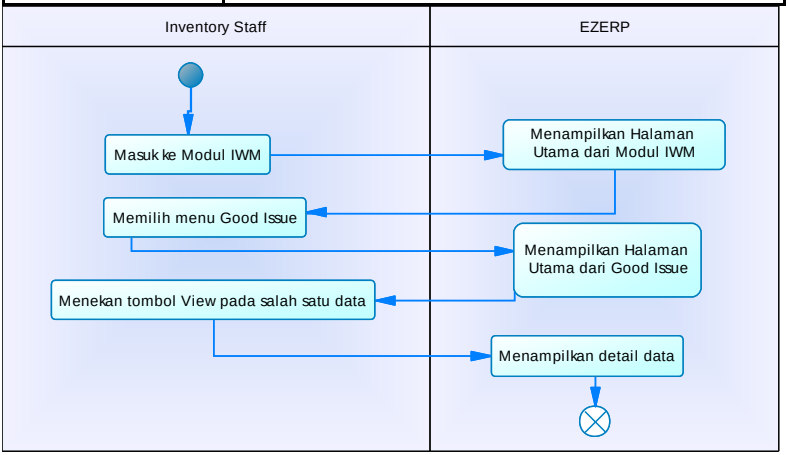
Gambar 3.3.22 Diagram aktivitas use case UC-010

3.2.5.11 UC-011 Melihat Data Permintaan Barang

Tabel 3.15 Spesifikasi Kasus Penggunaan Melihat Data Permintaan Barang

Nama	Melihat Data Permintaan Barang
Nomor	UC-011
Deskripsi	Kasus penggunaan ini digunakan untuk melihat seluruh data permintaan barang baik dari distribusi, produksi, maupun aset
Tipe	Fungsional
Aktor	Manajer Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman utaman dari *Good Issue*
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih menu Good Issue 4. Sistem menampilkan halaman utama Good Issue 5. Aktor memilih pilihan melihat pada data tertentu 6. Sistem menampilkan detail data tertentu

	7. Kasus penggunaan berakhir
Alur Alternatif	-



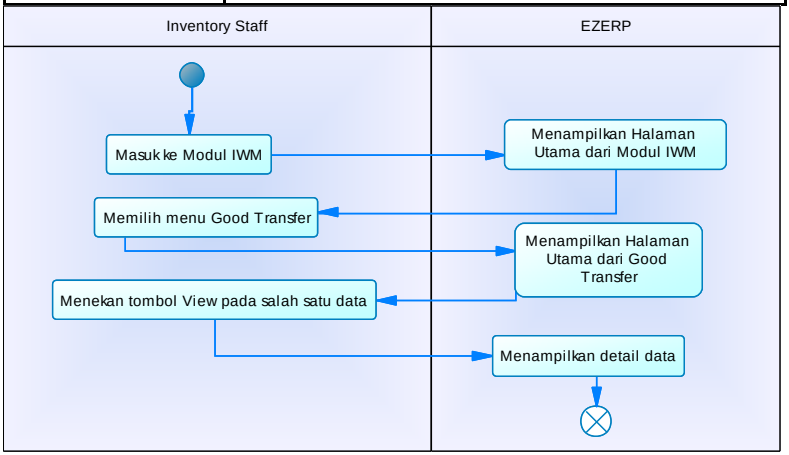
Gambar 3.3.23 Diagram aktivitas use case UC-011

3.2.5.12 UC-012 Melihat Data Pengiriman Barang di Luar Gudang

Tabel 3.16 Spesifikasi Kasus Penggunaan Melihat Data Pengiriman Barang di Luar Gudang

Nama	Melihat Data Pengiriman Barang di Luar Gudang
Nomor	UC-012
Deskripsi	Kasus penggunaan ini digunakan untuk melihat seluruh data pengiriman barang yang mengarah pengeluaran barang dari gudang
Tipe	Fungsional
Aktor	Manajer Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman utaman dari *Good Transfer*
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP

	3. Aktor memilih menu Good Transfer 4. Sistem menampilkan halaman utama Good Transfer 5. Aktor memilih pilihan melihat pada data tertentu 6. Sistem menampilkan detail data tertentu 7. Kasus penggunaan berakhir
Alur Alternatif	-



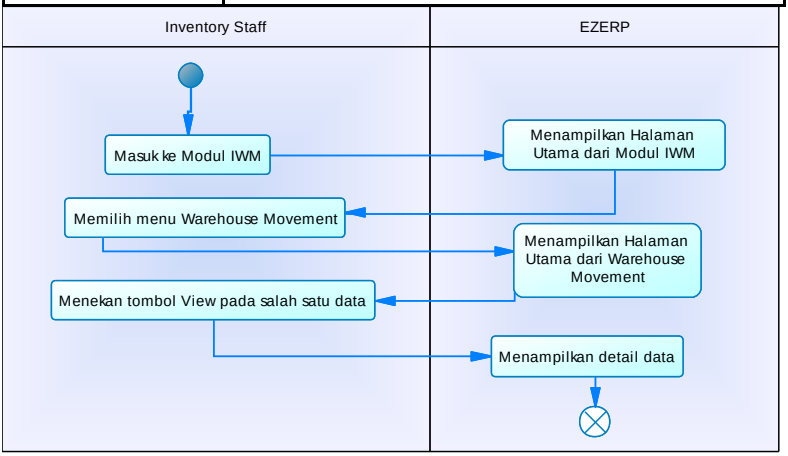
Gambar 3.3.24 Diagram aktivitas use case UC-012

3.2.5.13 UC-013 Melihat Data Perpindahan Barang di Dalam Gudang

Tabel 3.17 Spesifikasi Kasus Penggunaan Melihat Data Perpindahan Barang di Dalam Gudang

Nama	Melihat Data Perpindahan Barang di Dalam Gudang
Nomor	UC-013
Deskripsi	Kasus penggunaan ini digunakan untuk melihat seluruh data perpindahan barang yang terjadi di dalam gudang, baik dari truk ke dalam gudang maupun dari dalam gudang ke dalam truk, atau perpindahan dari satu lokasi ke lokasi lainnya di dalam gudang
Tipe	Fungsional
Aktor	Manajer Gudang

Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman utaman dari *Warehouse Movement*
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih menu Warehouse Movement 4. Sistem menampilkan halaman utama Warehouse Movement 5. Aktor memilih pilihan melihat pada data tertentu 6. Sistem menampilkan detail data tertentu 7. Kasus penggunaan berakhir
Alur Alternatif	-



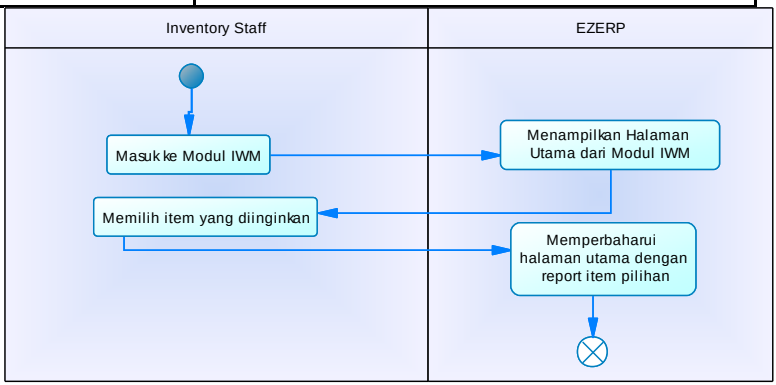
Gambar 3.3.25 Diagram aktivitas use case UC-013

3.2.5.14 UC-014 Melihat Report Jumlah Perpindahan Barang

Tabel 3.18 Spesifikasi Kasus Penggunaan Melihat Report Jumlah Perpindahan Barang

Nama	Melihat Report Jumlah Perpindahan Barang
Nomor	UC-014

Deskripsi	Kasus penggunaan ini digunakan untuk melihat report seluruh jumlah perpindahan di dalam gudang tercatat per hari
Tipe	Fungsional
Aktor	Staff Gudang, Manajer Gudang
Kondisi Awal	Pengguna sudah masuk ke sistem
Kondisi Akhir	Sistem menampilkan halaman utama dari modul IWM ERP
Alur Normal	1. Aktor membuka halaman utama modul IWM ERP 2. Sistem menampilkan halaman utama modul IWM ERP 3. Aktor memilih item yang akan dilihat reportnya 4. Sistem memperbaharui halaman dengan roport item pilihan 5. Kasus penggunaan berakhir
Alur Alternatif	-



Gambar 3.3.26 Diagram aktivitas use case UC-014

3.2.6 **Perancangan Basis Data Terdistribusi**

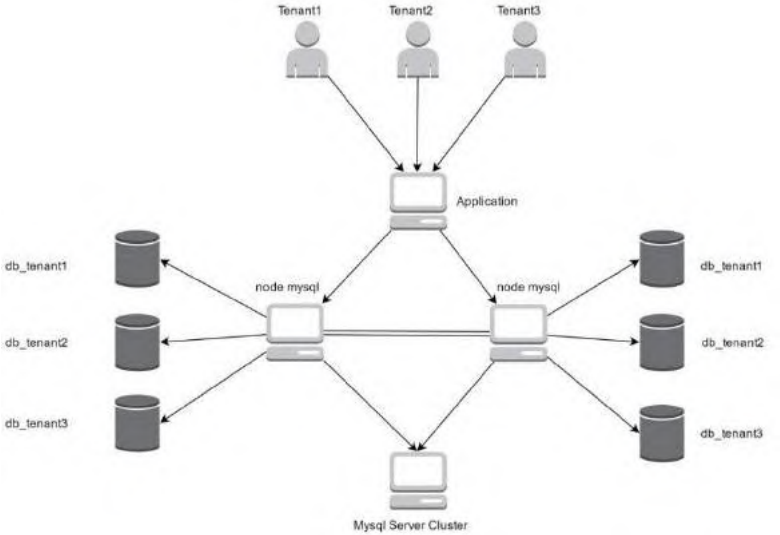
Perangkat lunak yang akan dibangun menggunakan basis data terdistribusi yang dijabarkan pada gambar 3.4.

Pada gambar 3.4 menjelaskan bahwa perusahaan 1, perusahaan 2, dan perusahaan 3 dapat menggunakan ERP secara bersamaan. Setiap

perusahaan yang terdaftar dalam sistem ERP akan menyimpan data dalam 2 *database server* dimana akan menggunakan sebuah *Management Server* yang bertugas dalam me-*manage* replikasi pada kedua buah *server* untuk meningkatkan *availability* ketika ada salah satu *database server* yang mengalami *down*, system ERP masih tetap berjalan.

3.2.7 **Perancangan multi-tenancy**

Pada bagian ini akan dijelaskan mengenai perancangan multi-tenancy *yang* terdapat pada sistem ERP 2016. Secara rinci mengenai multi-tenancy dijabarkan sebagai berikut:



Gambar 3.27 Perancangan Multi-tenancy

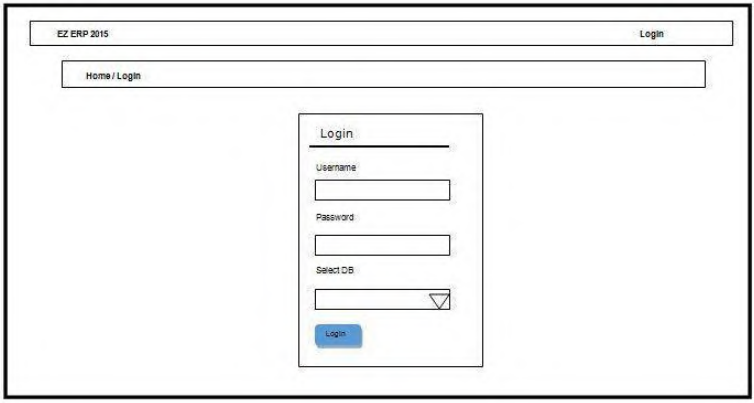
Gambar 3.2.2 menerangkan bagaimana alur multi-tenancy yang terjadi pada sistem ERP. Pembagian node mysql sudah dijelaskan bada sub-bab 3.2.1 yaitu dengan cara clustering menjadi 2 database yang berbeda. Basis data terdistribusi merupakan penunjang dari multi-

tenancy, yang berfungsi untuk mengatur session setiap perusahaan yang akan mengakses sistem ERP.

3.2.8 **Perancangan RBAC (Role Base Acces Control)**

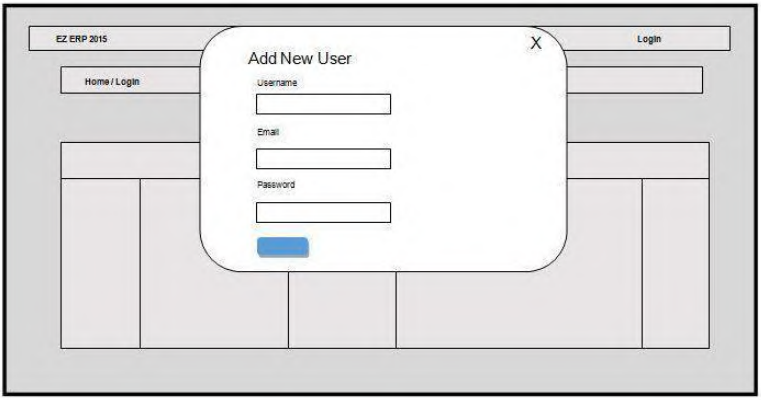
Pada bagian ini akan dijelaskan mengenai perancangan RBAC *yang* terdapat pada sistem ERP 2016. Secara rinci mengenai RBAC dijabarkan sebagai berikut

1. Perancangan antarmuka login



Gambar 3.28 Perancangan antarmuka Login

2. Perancangan antarmuka add user



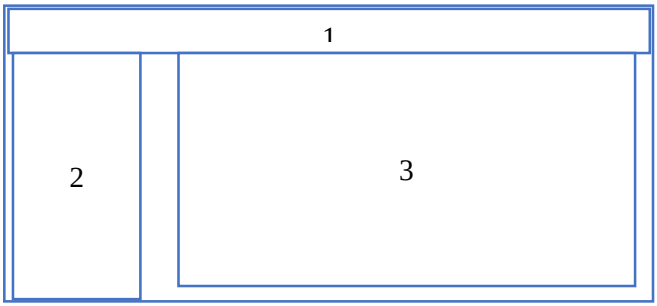
Gambar 3.29 Perancangan antarmuka add user

3.2.9 **Perancangan Tampilan**

Pada sub-bab ini membahas perancangan antarmuka yang akan digunakan dalam aplikasi ERP khususnya modul IWM. Ilustrasi perancangan ditunjukkan sebagai gambar disertai keterangan objek-objek yang ada di dalamnya.

3.2.9.1 Perancangan Antarmuka Halaman Dashboard

Rancangan antarmuka halaman dashboard ditunjukkan pada gambar di bawah ini. Pada halaman ini ditujukan untuk pengguna agar dapat memonitor keseluruhan modul IWM.



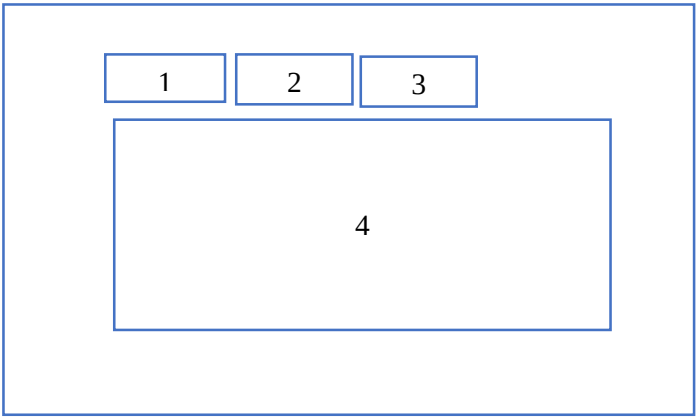
Gambar 3.30 Rancangan Antarmuka Halaman *Dashboard*

Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. Menu utama dari aplikasi
2. Menu dari modul IWM
3. Report perpidahan barang pada gudang dan juga sebagai tampilan untuk seluruh proses yang ada pada modul

3.2.9.2 Perancangan Antarmuka Halaman Utama Setiap SubModul

Rancangan antarmuka halaman utama untuk setiap submodul ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat melihat seluruh data yang ada pada submodul tersebut dan melakukan proses pengelolaan baik itu membuat, menyunting, menghapus, melihat dan memproses data, kemudian memasukkan data ke dalam basis data.



Gambar 3.31 Rancangan Antarmuka Halaman *Utama Setiap SubModul*

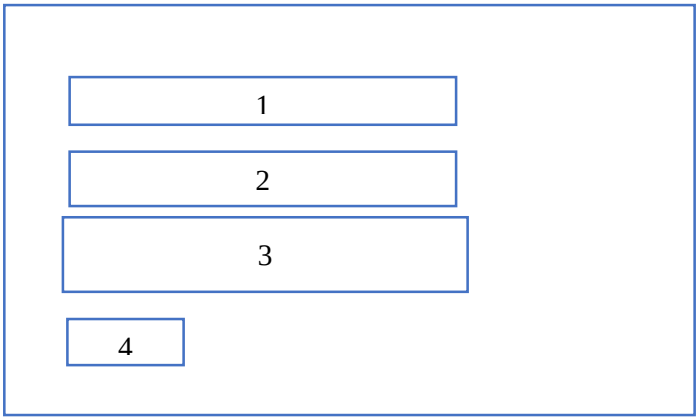
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. Tombol untuk *Create Data*
2. Tombol untuk *Update Data*

3. Tombol untuk *Delete Data*
4. Halaman yang menampilkan seluruh data pada suatu submodul.

3.2.9.3 Perancangan Antarmuka Warehouse

Rancangan antarmuka halaman *Warehouse* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat warehouse baru dan kemudian memasukkan data ke dalam basis data.



Gambar 3.32 Rancangan Antarmuka Halaman *Warehouse*

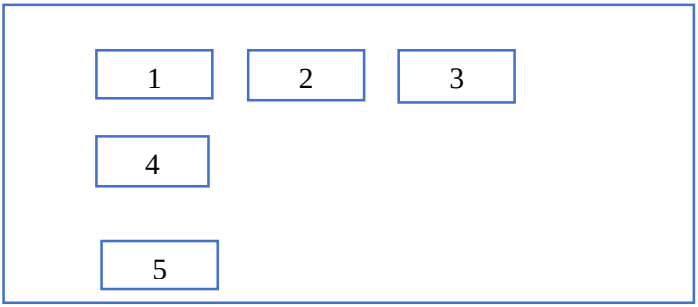
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input teks* untuk *Warehouse name.*
2. *Input* teks untuk *Warehouse Location.*
3. *Input* teks untuk *Description.*
4. Tombol *Create*.

   Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman utama *warehouse.*

3.2.9.4 Perancangan Antarmuka Inventory

Rancangan antarmuka halaman *Inventory* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat Inventory baru yang berada di dalam *warehouse* dan kemudian memasukkan data ke dalam basis data.



Gambar 3.33 Rancangan Antarmuka Halaman *Inventory*

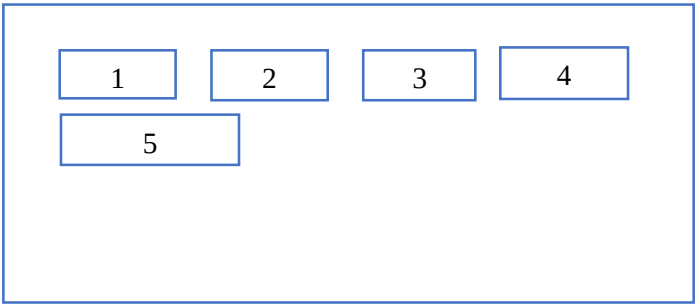
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input dropdown* untuk Warehouse *ID.*
2. *Input* teks untuk *Inventory Name.*
3. *Input* teks untuk Inventory Code.
4. *Input* teks untuk *Inventory Location*
5. Tombol *Create*.

   Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Inventory*.

3.2.9.5 Perancangan Antarmuka Storage

Rancangan antarmuka halaman *Storage* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat *storage* baru yang berada di dalam *inventory* dan kemudian memasukkan data ke dalam basis data.



Gambar 3.34 Rancangan Antarmuka Halaman *Storage*

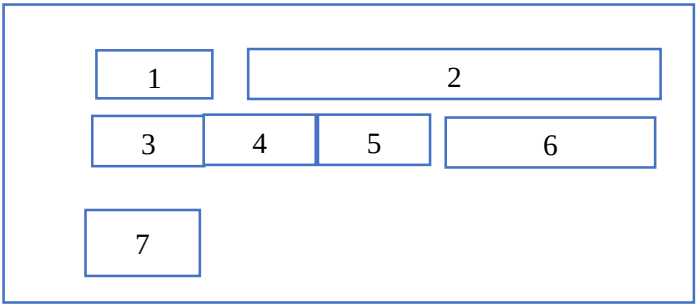
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input dropdown* untuk *Inventory ID.*
2. *Input* teks untuk *Storage Name.*
3. *Input* teks untuk *Storage Type.*
4. *Input checkbox* untuk *Storage Status.*
5. Tombol *Create*.

   Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Storage*.

3.2.9.6 Perancangan Antarmuka Halaman Storage Detail

Rancangan antarmuka halaman *Storage Detail* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat *storage detail* baru yang berada di dalam *storage* dan kemudian memasukkan data ke dalam basis data.



Gambar 3.35 Rancangan Antarmuka Halaman *Storage*

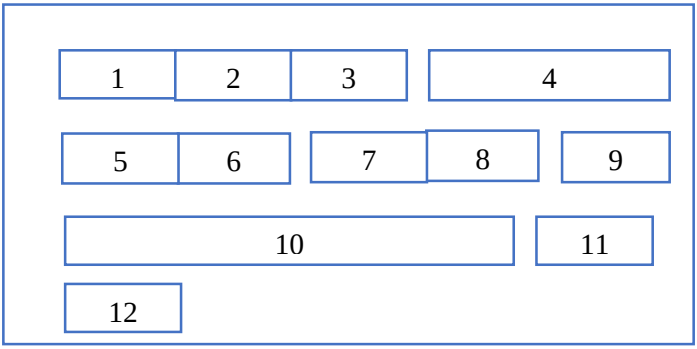
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input dropdown* untuk *Storage ID.*
2. *Input* teks untuk *Storage Detail Name.*
3. *Input* teks untuk *Row Number.*
4. *Input* teks untuk *Shelf Number.*
5. *Input* teks untuk *Stack Number.*
6. *Input* teks untuk *Storage Section.*
7. Tombol *Create*.

   Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Storage Detail.*

3.2.9.7 Perancangan Antarmuka Halaman Bin

Rancangan antarmuka halaman *Bin* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat *bin* baru yang berada di dalam *storage detail* dan kemudian memasukkan data ke dalam basis data.



\ **Gambar 3.36 Rancangan Antarmuka Halaman *Bin***

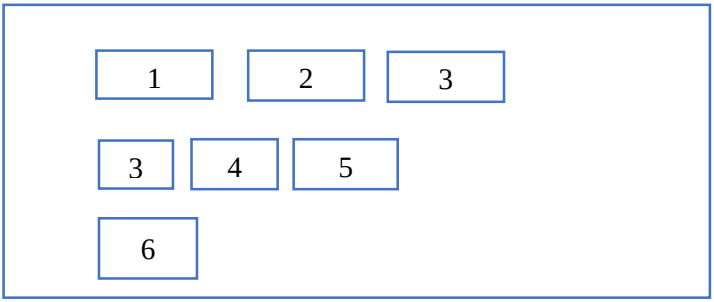
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input dropdown* untuk *Storage Detail ID.*
2. *Input* teks untuk *Shelf Number.*
3. *Input* teks untuk *Stack Number.*
4. *Input* teks untuk *Bin Name.*
5. *Input* teks untuk *Volume Uom.*
6. *Input* teks untuk *Max Volume.*
7. *Input* teks untuk *Weight Uom.*
8. *Input* teks untuk *Max Weight.*
9. *Input checkbox* untuk *Bin Status.*
10. *Input* teks untuk Item ID.
11. *Input* teks untuk *Max Item Placed.*
12. Tombol *Create*.

    Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Bin.*

3.2.9.8 Perancangan Antarmuka Halaman Quant

Rancangan antarmuka halaman *Quant* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat *quant* baru yang berada di dalam *bin* dan kemudian memasukkan data ke dalam basis data.



Gambar 3.37 Rancangan Antarmuka Halaman *Quant*

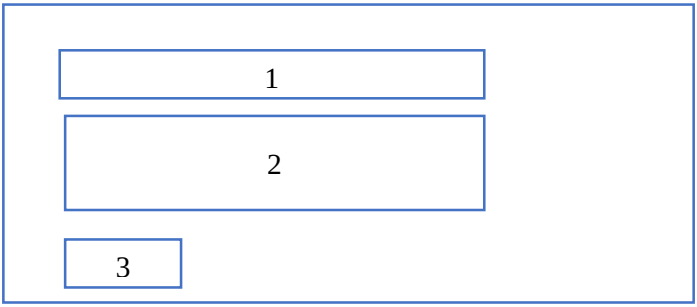
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input dropdown* untuk *Bin ID.*
2. *Input dropdown* untuk *Sales ID.*
3. *Input dropdown* untuk *Production ID.*
4. *Input dropdown* untuk *Item ID.*
5. *Input* teks untuk *Item Quantity*.
6. *Input* teks untuk *Status.*
7. Tombol *Create*.

   Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Quant.*

3.2.9.9 Perancangan Antarmuka Halaman Item Type.

Rancangan antarmuka halaman *Item Type* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat *quant* baru yang berada di dalam perusahaan dan kemudian memasukkan data ke dalam basis data.



Gambar 3.38 Rancangan Antarmuka Halaman *Item Type*

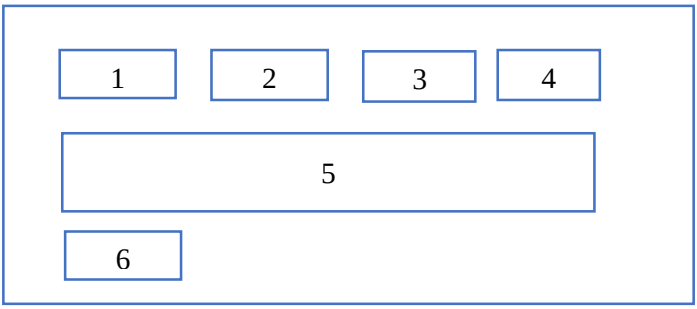
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input* teks untuk *Item Type Name.*
2. *Input* teks untuk *Description.*
3. Tombol *Create*.

   Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Item Type.*

3.2.9.10 Perancangan Antarmuka Halaman Item Master

Rancangan antarmuka halaman *Item Master* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat *Item Master* untuk setiap *Item Type* yang ada di Perusahaan dan kemudian memasukkan data ke dalam basis data.



Gambar 3.39 Rancangan Antarmuka Halaman *Item Master*

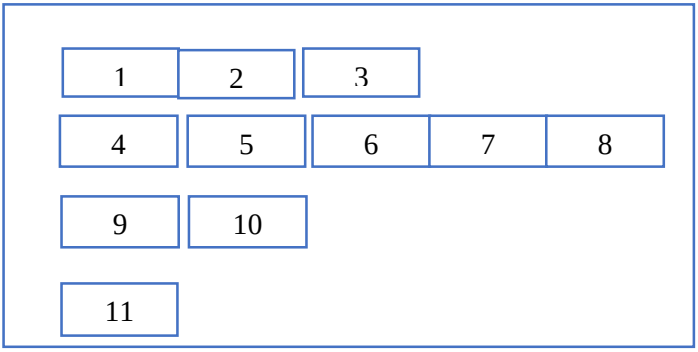
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input* teks untuk *Item Name.*
2. *Input dropdown* untuk *Item Type ID.*
3. *Input dropdown* untuk *UoM ID.*
4. *Input* teks untuk *Item Weight.*
5. *Input date* untuk *Description.*
6. Tombol *Create*.

   Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Item Master*.

3.2.9.11 Perancangan Antarmuka Halaman Item Detail

Rancangan antarmuka halaman *Item Detail* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat *Item Detail* untuk setiap Item Master yang ada di Perusahaan dan kemudian memasukkan data ke dalam basis data.



Gambar 3.40 Rancangan Antarmuka Halaman *Item Detail*

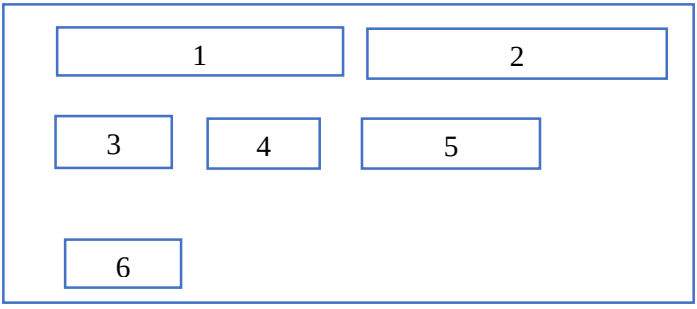
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input dropdown* untuk *Purchase Order ID.*
2. *Input dropdown* untuk *Purchase Order Line ID.*
3. *Input dropdown* untuk *Production Order ID.*
4. *Input dropdown* untuk *Item ID.*
5. *Input* teks untuk *Item Name.*
6. *Input* teks untuk *Cost.*
7. *Input* teks untuk *Price.*
8. *Input dropdown* untuk *Currency ID.*
9. *Input* teks untuk *Quantity in Inventory.*
10. *Input* teks untuk *Buy Quantity.*
11. Tombol *Create*.

    Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Item Detail.*

3.2.9.12 Perancangan Antarmuka Halaman Stock

Rancangan antarmuka halaman *Stock* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat *Stock* untuk setiap *Item Master* yang ada di Perusahaan dan kemudian memasukkan data ke dalam basis data.



Gambar 3.41 Rancangan Antarmuka Halaman *Stock*

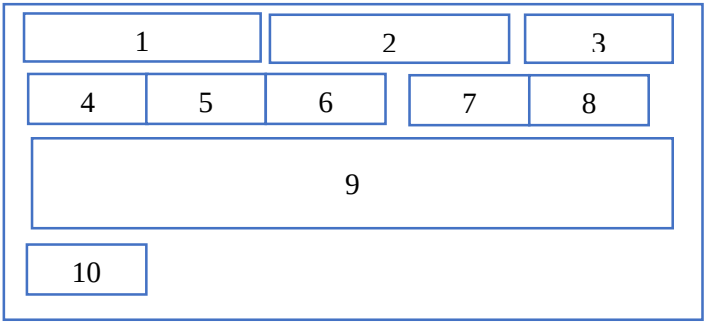
Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input dropdown* untuk *Item Master ID.*
2. *Input dropdown* untuk *Item Type ID.*
3. *Input* teks untuk *Current Quantity*.
4. *Input* teks untuk *Quantity Needed.*
5. *Input* teks untuk Max Quantity.
6. Tombol *Create*.

Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Stock.*

3.2.9.13 Perancangan Antarmuka Halaman Good Receipt

Rancangan antarmuka halaman *Good Receipt* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat transaksi *Good Receipt* yang terjadi di Perusahaan. Dan kemudian memasukkan data ke dalam basis data.



Gambar 3.42 Rancangan Antarmuka Halaman *Good Receipt*

Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input* teks untuk *Receipt Name.*
2. *Input date* untuk *Document Date.*

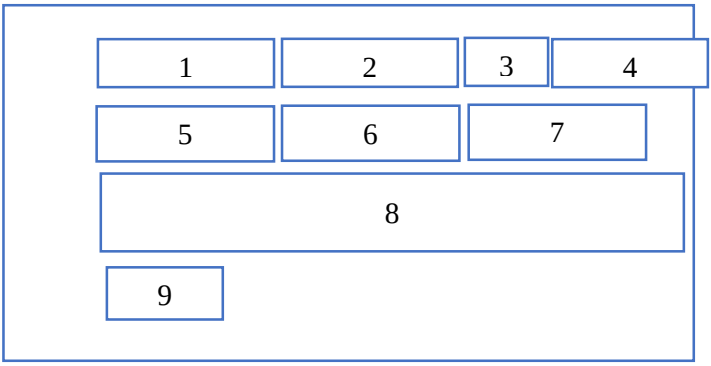
   Date ini mencatat tanggal saat pembuatan data receipt ini.
3. *Input dropdown* untuk *Receipt Status.*
4. *Input dropdown* untuk *From Purchase ID.*
5. *Input dropdown* untuk *From Customer ID.*
6. *Input dropdown* untuk *From Production ID.*

7. *Input dropdown* untuk *To Inventory ID.*
8. *Input dropdown* untuk *To Production ID.*
9. *Input* teks untuk Detail Receipt.
10. Tombol *Create.*

    Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Good Receipt.*

3.2.9.14 Perancangan Antarmuka Halaman Good Issue

Rancangan antarmuka halaman *Good Issue* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat transaksi *Good Issue* yang terdapat di Perusahaan dan kemudian memasukkan data ke dalam basis data.



Gambar 3.43 Rancangan Antarmuka Halaman *Good Issue.*

Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input* teks untuk *Issue Name.*
2. *Input date* untuk *Issue Date.*

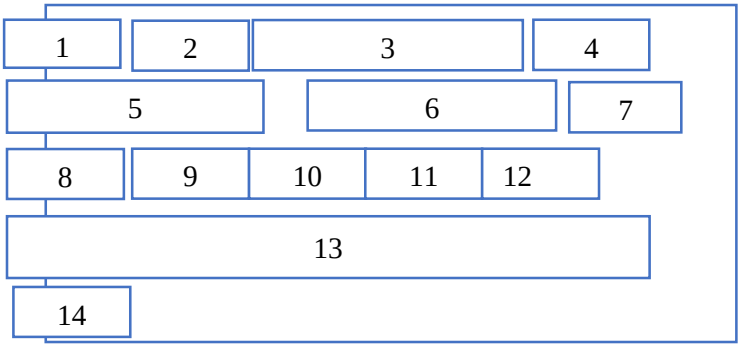
   Input ini mencatat tanggal saat issue ini di proses.
3. *Input* teks untuk *Issue Type.*
4. *Input dropdown* untuk *Issue Status.*
5. *Input dropdown* untuk *To Sales ID.*
6. *Input dropdown* untuk *To Production ID.*

7. *Input dropdown* untuk *To Inventory ID.*
8. *Input* teks untuk *Detail Issue.*
9. Tombol *Create*.

Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Good Issue*.

3.2.9.15 Perancangan Antarmuka Halaman Transfer Posting

Rancangan antarmuka halaman *Transfer Posting* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat transaksi *Transfer Posting* yang terdapat di Perusahaan dan kemudian memasukkan data ke dalam basis data.



Gambar 3.44 Rancangan Antarmuka Halaman *Transfer Posting*.

Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input dropdown* untuk *Issue ID.*
2. *Input* dropdown untuk *Purchase ID.*
3. *Input* teks untuk *Transfer Name.*
4. *Input teks* untuk *Batch.*
5. *Input date* untuk *Date Send*.

Date Send adalah waktu saat transfer ini dibuat.

6. *Input date* untuk *Date Shipped.*

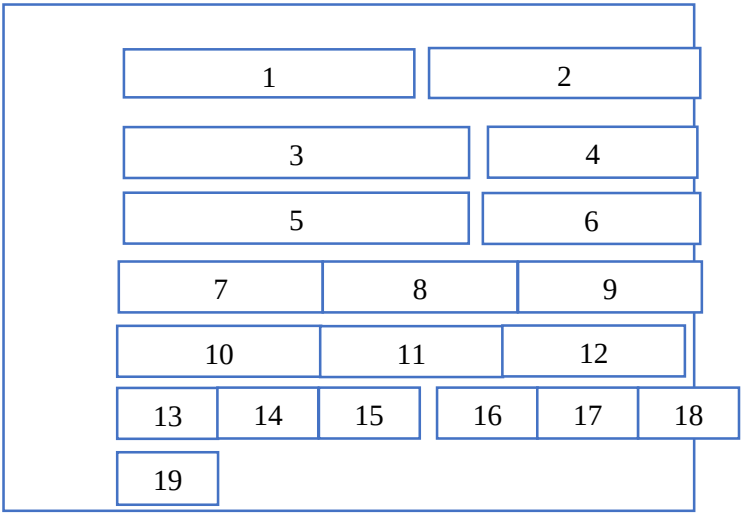
Date Shipped adalah waktu saat transfer ini selesai dikerjakan.

7. *Input dropdown* untuk *Transfer Status*
8. *Input dropdown* untuk *From Inventory ID.*
9. *Input dropdown* untuk *To Sales ID.*
10. *Input dropdown* untuk *To Deliver ID.*
11. *Input dropdown* untuk *To Production ID.*
12. *Input dropdown* untuk *To Inventory ID.*
13. *Input* teks untuk *Detail Transfer*.
14. Tombol *Create*.

Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Transfer Posting.*

3.2.9.16 Perancangan Antarmuka Warehouse Movement

Rancangan antarmuka halaman *Warehouse Movement* ditunjukkan oleh gambar di bawah ini. Pada halaman ini, pengguna dapat membuat *Warehouse Movement* untuk setiap perpindahan barang yang terdapat di gudang setiap hari dan kemudian memasukkan data ke dalam basis data.



Gambar 3.45 Rancangan Antarmuka Halaman *Warehouse Movement.*

Berikut penjelasan masing-masing nomor yang ada di dalam masing-masing kotak:

1. *Input dropdown* untuk *Receipt Detail ID.*
2. *Input dropdown* untuk *Issue Detail ID.*
3. *Input* teks untuk *Movement Name.*
4. *Input date* untuk *Date Taken.*

   Date Taken adalah waktu yang dicatat saat melakukan proses *movement.*
5. *Input dropdown* untuk *Item ID*
6. *Input dropdown* untuk *UoM ID.*
7. *Input* teks untuk *Item Quantity*.
8. *Input* teks untuk *Quantity in Inventory.*
9. *Input* teks untuk *Total Quantity.*
10. *Input dropdown* untuk *From Production ID.*
11. *Input dropdown* untuk *To Production ID.*
12. *Input dropdown* untuk *To Sales ID.*
13. *Input dropdown* untuk *From Storage ID.*
14. *Input dropdown* untuk *From Storage Detail ID.*
15. *Input dropdown* untuk *From Bin ID.*
16. *Input dropdown* untuk *To Storage ID.*
17. *Input dropdown* untuk *To Storage Detail ID.*
18. *Input dropdown* untuk *To Bin ID.*
19. Tombol *Create*.

    Jika tombol ini ditekan maka aplikasi akan menyimpan data yang dimasukkan di dalam *input*. Dan mengarahkan ke halaman *Warehouse Movement.*

BAB IV
IMPLEMENTASI SISTEM

Bab ini membahas implementasi dari perancangan sistem ERP 2015. Di dalamnya mencakup penjelasan lingkungan pengembangan sistem serta proses implementasi *distributed database, RBAC, multi-tenancy,* dan antarmuka pengguna.

4.1 Lingkungan Pengembangan Sistem

Lingkungan pengembangan sistem yang digunakan untuk mengembangkan Tugas Akhir ini dilakukan pada lingkungan dan kakas sebagai berikut.

1. *Database* yang digunakan pada server adalah *MySQL Cluster*.
2. 2 PC untuk *server database* menggunakan Ubuntu 12.04.
3. 1 PC untuk manajemen *server database* menggunakan Ubuntu 12.04.
4. 1 PC untuk pengembangan *User* menggunakan Intel® Core™ i3-2120 @3.30GHz , RAM 4GB dengan Sistem Operasi Windows 8.1 Enterprise x64.
5. 1 PC untuk uji coba menggunakan Intel® Core™ i3-2120 @3.30GHz , RAM 4GB dengan Sistem Operasi Windows 8.1 Enterprise x64.
6. StarUML untuk pembuatan diagram, Power Designer 12.5, Sublime sebagai teks editor, Pencil untuk pembuatan desain antarmuka, Power Designer untuk pembuatan CDM dan PDM.
7. Mozilla Firefox 46.0.1 sebagai antarmuka untuk pengujian aplikasi klien.

4.2 Implementasi Distributed Database

Pada bagian ini akan dijelaskan mengenai implementasi *distributed database* yang terdapat pada sistem ERP 2016. Secara rinci mengenai implementasi *distributed database* dijabarkan sebagai berikut:

4.2.1 Instalasi Data dan SQL node pada node1 dan node2

Pada implementasi instalasi dan sql node pada node1 dan node2, dilakukan langkah-langkah sebagai berikut:

1. Membuat grup MySQL pengguna baru, kemudian menambah user MySQL. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.1.

```
shell> groupadd mysql
shell> useradd-g mysql mysql
```

Kode Sumber 4.2.1 Membuat grup MySQL pengguna baru dan menambah user MySQL

2. Mengubah lokasi ke dalam direktori yang berisi file yang telah di download, kemudian mengubah arsip dan menciptakan symlink ke dalam direktori mysql yang bernama "mysql". Hal yang perlu diperhatikan adalah, file yang sebenarnya dan nama direktori bervariasi sesuain dengan jumlah cluster versi MySQL. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.2.

```
shell> cd/usr/local
shell>/usr/local$ tar xzvf mysql-cluster-gpl-7.1.34-
      linux-x86_64-glibc23
shell>ln-s/usr/local/mysql-cluster-gpl-7.1.34-linux-
      x86-glibc23/usr/local/mysql
shell> export PATH = $ PATH:/usr/local/mysql/bin
shell>    echo    "export    PATH=\$PATH:/usr/local/
      mysql/bin"> /etc/bash.bashrc
```

Kode Sumber 4.2.2 Mengubah lokasi ke dalam direktori yang berisi file yang telah didownload, mengubah arsip dan menciptakan symlink ke dalam direktori mysql.

3. Mengubah lokasi ke dalam direktori mysql dan menjalankan script untuk menciptakan *database system*. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.3.

```
shell> cd mysql
shell>. / scripts / mysql_install_db-user = mysql
```

Kode Sumber 4.2.3 Mengubah lokasi ke direktori mysql.

4. Mengatur izin yang diperlukan oleh server MSQL. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.4.

```
shell> chown-R root.
shell> chown-R mysql data
shell> chgrp-R mysql.
```

Kode Sumber 4.2.4 Mengatur izin yang diperlukan oleh server MSQL.

5. Menyalin *script startup MySQL* ke direktori yang sesuai, mengubah menjadi *executable*, dan memulai ketika sistem beroperasi. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.5.

```
shell> cp support-files/mysql.server / etc / init.d /
mysql
shell> chmod + x / etc / init.d / mysql
shell> update-rc.d mysql defaults
```

Kode Sumber 4.2.5 Menyalin *script startup MySQL* ke direktori yang sesuai, mengubah menjadi *executable*, dan memulai ketika sistem beroperasi

4.2.2 Pemasangan Node Manajemen pada node03

Pemasangan node manajemen memerlukan manajemen server *MySQL Cluster* (ndb_mgmd), diasumsikan bahwa mysql-cluster-gpl-7.1.5-linux-i686-glibc23.tar.gz telah ditempatkan di /var / tmp. Untuk memasang ndb_mgmd dan ndb_mgm pada *host Cluster*, sistem sebagai *root* melakukan langkah-langkah sebagai berikut:

1. Mengubah lokasi ke dalam direktori / var / tmp direktori, dan mengekstrak ndb_mgm dan ndb_mgmd dari arsip ke direktori yang sesuai seperti / usr / local / bin. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.6.

```
shell> cd/usr/local
```

```
shell> tar-zxvf mysql-cluster-gpl-7.1.34-linux-x86-
glibc23.tar.gz
shell>  cd  /  usr/local/mysql-cluster-gpl-7.1.34-
linux-x86-glibc235
shell> cp bin / ndb mgm */usr/local/bin
```

Kode Sumber 4.2.6 Mengubah lokasi ke dalam direktori / var / tmp direktori, mengekstrak ndb_mgm dan ndb_mgmd dari arsip ke direktori yang sesuai seperti / usr / local / bin.

2. Mengubah lokasi ke dalam direktori tempat file disalin, kemudian dieksekusi. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.7.

```
shell> cd/usr/local/bin
shell> chmod +x ndb_mgm*
```

Kode Sumber 4.2.7 Mengubah lokasi ke dalam direktori tempat file disalin, kemudian dieksekusi.

4.2.3 Konfigurasi Manajemen Node

Konfigurasi pada manajemen node dilakukan dengan langkah-langkah sebagai berikut:

1. Membuat direktori tempat file konfigurasi ditemukan kemudian membuat file itu sendiri. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.8.

```
shell> mkdir / var / lib / mysql-cluster
shell> cd / var / lib / mysql-cluster
vi config.ini
```

Kode Sumber 4.2.8 Membuat direktori tempat file konfigurasi ditemukan kemudian membuat file itu sendiri.

2. Mengatur file “config.ini”. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.9.

```
[ndbd default]
NoOfReplicas=2
DataMemory=10G
IndexMemory=2G
MaxNoOfAttributes=10000
MaxNumberOfTables=2500
MaxOfOrderedIndexes=4086
MaxOfConcurrentOperations=250000
MaxOfConcurrentOperations=250000
[tcp default]
[ndb_mgmd]
hostname=10.151.64.182
datadir=/var/lib/mysql-cluster
[ndbd]
hostname=10.151.64.182
datadir=/usr/local/mysql/data
[ndbd]
hostname=10.151.64.203
datadir=/usr/local/mysql/data
[mysqld]
MaxNoOfAttributes=10000
hostname=10.151.64.182
[mysqld]
MaxNoOfAttributes=10000
hostname=10.151.64.203
```

Kode Sumber 4.2.9 Mengatur file "config.ini".

4.2.4 **Konfigurasi Data dan SQL Node**

Konfigurasi data dan SQL Node dilakukan dengan cara mengedit file my.cnf pada direktori /etc/. Untuk setiap data node dan SQL node yang diatur pada my.cnf. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.10.

```
[client]
port=3306
socket=/tmp/mysql.sock
[mysqld]
port=3306
socket-/tmp/mysql.sock
ndbcluster
ndb-connectstring=10.151.64.181
[mysql_cluster]
ndb-connectstring=10.151.64.181
```

Kode Sumber 4.2.10 Data dan SQL Node

4.2.5 **Memulai *MySQL Cluster***

Setiap proses *node cluster* harus dimulai secara terpisah. Manajemen *node* harus dimulai terlebih dahulu, kemudian *node* data. Pada setiap node SQL dilakukan langkah sebagai berikut :

1. Pada node03 (host manajemen), untuk memulai proses manajemen node dari shell sistem dilakukan perintah berikut. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.11.

```
shell>      ndb_mgmd      -f      /var/lib/mysql-
cluster/config.ini-configdir=/var/lib/mysql-
cluster/
```

Kode Sumber 4.2.11 Memulai proses manajemen *node*

Jika berhasil akan muncul tampilan yang ditunjukkan oleh gambar 4.2.1.

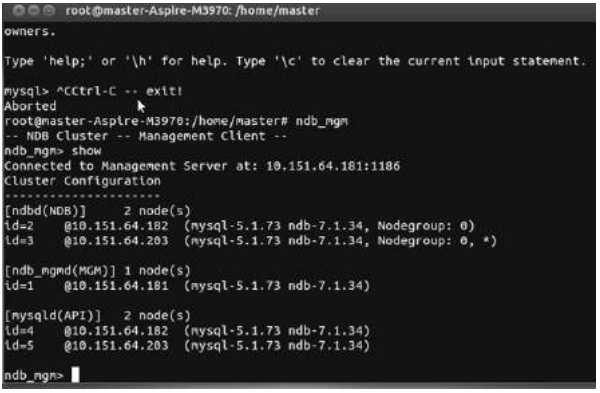
Gambar 4.1 Memulai Proses Manajemen *Node*

2. Jalankan perintah untuk memulai ndbd dan proses *mysql server* pada masing-masing Data/*host* SQL. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.12.

```
shell> / usr / local / mysql / bin / ndbd
```

Kode Sumber 4.2.12 Memulai proses ndbd dan proses *mysql server*.

Jika berhasil akan keluar tampilan seperti Gambar 4.2.2.



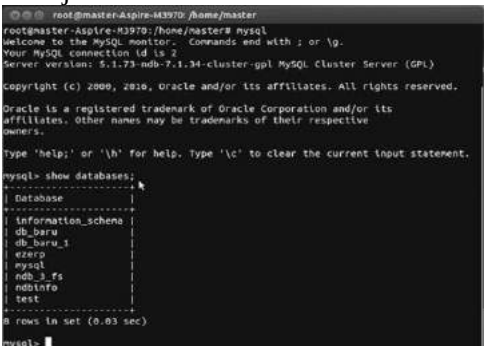
Gambar 4.2 ndbd dan proses mysql server dapat dimulai.

3. Mengaktifkan mysql pada data node. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.2.13.

```
shell> / etc / init.d / mysql start
```

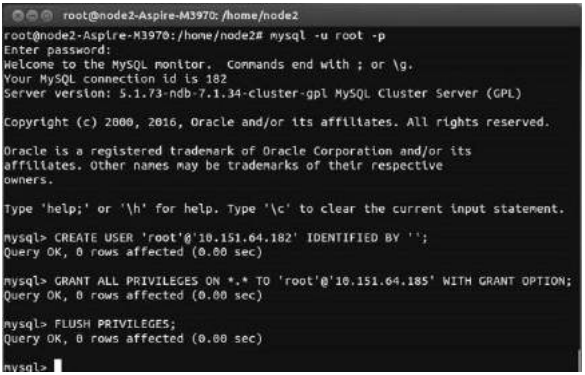
Kode Sumber 4.2.13 Memulai proses ndbd dan proses mysql server.

Jika berhasil yang maka akan terlihat tampilan yang ditunjukkan oleh Gambar 4.2.3.



Gambar 4.3 Mysql pada data node aktif

4. Menguji pada client node dengan perintah ndb_mgm. Jika berhasil yang maka akan terlihat tampilan yang ditunjukkan oleh Gambar 4.2.4.



Gambar 4.4 Data Node Saling Terkoneksi

4.3 Implementasi RBAC (Role Base Acces Control)

Pada bagian ini akan dijelaskan mengenai implementasi *RBAC* yang terdapat pada sistem ERP 2016. Secara rinci mengenai implementasi *RBAC* dijabarkan sebagai berikut:

4.3.1 Membuat Tabel Pengguna

Pembuatan tabel pengguna dilakukan dengan menjalankan perintah "*yii migrate*" pada folder aplikasi. Jika berhasil yang maka akan muncul keterangan "*migrate successfully*".

4.3.2 Membuat 4 Tabel Autentifikasi RBAC dan Tabel Pengguna

Pada tahap ini dibutuhkan 4 tabel autentifikasi yang terdiri dari:

1. Tabel *Item*
2. Tabel *Child*
3. Tabel *Assigment*
4. Tabel *Rule*

Cara untuk membuat tabel-tabel tersebut yaitu dengan menjalankan perintah yang ditunjukkan pada Kode Sumber 4.3.1.

```
      Yii migrate --
migrationPath=@yii/rbac/migrations
```

Kode Sumber 4.3.1 Generate Tabel Autentifikasi

Jika berhasil yang maka akan terbuat 4 tabel pada database, yaitu tabel ***auth_assigment***, tabel ***auth_item***, tabel ***auth_item_child*** dan tabel ***auth_rule***.

4.3.3 Membuat Modul *Admin*

Proses ini bertujuan untuk meletakkan konfigurasi pengguna dan masing-masing model dari 4 tabel autentifikasi. Kemudian dilakukan konfigurasi pada folder config file `web.php`. Terdapat 3 kofigurasi yaitu:

1. Admin : digunakan sebagai akses ke modul admin
2. Auth Manager : sebagai autentifikasi di yii2 dan mengatur role default sebagai *guest*
3. Session Time Out : megatur durasi time out session selama 5 menit atau 300 detik.

Masing –masing konfigurasi tersebut ditunjukkan pada Kode Sumber 4.3.2.

```
$config = [
      //…
      'modules' => [
            'admin' => [
                  'class'=>'app\modules\admin\AdminModule,
            ],
      ],
      'components' => [
            'authManager' => [
                  'class' => 'yii\rbac\DbManager',
                  'defaultRoles' => ['guest'],
            ],
            'session' => [
                  'timeout' => 300,
            ],
            //…
      ],
      //…
];
```

Kode Sumber 4.3.2 Pembuatan modul admin dan konfigurasi autentifikasi

4.3.4 Membuat Model Tabel Autentifikasi, *Controller* dan *View* Pengguna

Pada tahap ini dilakukan *generate* kelas model dari masing-masing tabel autentifikasi pada modul *admin* yang diperlukan pada tahap sebelumnya. Kemudian ditambahkan *generate* kelas *controller* dan *view* pada tabel pengguna.Proses *generate* ini menggunakan *yii generator* yang telah disediakan oleh *framework yii*.

4.3.5 Menambahkan Kode pada Kelas *usercontroller*

Pada modul *admin*, file `usercontroller.php` ditambahkan kode fungsi untuk semua tabel autentifikasi. Masing-masing fungsi ditunjukkan pada Kode Sumber 4.3.3, Kode Sumber 4.3.4, Kode Sumber 4.3.5.

```
public function actionAuthItem()
        {
           $auth = Yii::$app->authManager;

          // menambahkan akses sebagai admin ke tabel
    auth_item
              $admin = $auth->createPermission('admin');
              $admin->description = 'Allow user to access
       all page';
              $auth->add($admin);

              // menambahkan akses sebagai asset management
       manajer ke tabel auth_item
              $am_manager = $auth->createPermission('am-
       manager');
              $am_manager->description = 'Allow user as
       Asset Management Manager';
              $auth->add($am_manager);

              // menambahkan akses sebagai asset management
       staff ke tabel auth_item
              $am_staff = $auth->createPermission('am-
       staff');
              $am_staff->description = 'Allow user as Asset
       Management Staff';
              $auth->add($am_staff);

              // menambahkan akses sebagai account payable
       manajer ke tabel auth_item
              $ap_manager = $auth->createPermission('ap-
       manager');
              $ap_manager->description = 'Allow user as
       Account Payable Manager';
              $auth->add($ap_manager);
```

```
        // menambahkan akses sebagai account payable
staff ke tabel auth_item
        $ap_staff   =   $auth->createPermission('ap-
staff');
        $ap_staff->description  =  'Allow  user  as
Account Payable Staff';
        $auth->add($ap_staff);

        //   menambahkan   akses   sebagai   account
receivable manajer ke tabel auth_item
        $ar_manager  =  $auth->createPermission('ar-
manager');
        $ar_manager->description  =  'Allow  user  as
Account Receivable Manager';
        $auth->add($ar_manager);

        //   menambahkan   akses   sebagai   account
receivable staff ke tabel auth_item
        $ar_staff   =   $auth->createPermission('ar-
staff');
        $ar_staff->description  =  'Allow  user  as
Account Receivable Staff';
        $auth->add($ar_staff);

        // menambahkan akses sebagai finance manajer
ke tabel auth_item
        $fi_manager  =  $auth->createPermission('fi-
manager');
        $fi_manager->description  =  'Allow  user  as
Finance Manager';
        $auth->add($fi_manager);

        // menambahkan akses sebagai finance staff ke
tabel auth_item
        $fi_staff   =   $auth->createPermission('fi-
staff');
        $fi_staff->description  =  'Allow  user  as
Finance Staff';
        $auth->add($fi_staff);

        //      menambahkan      akses      sebagai
accounting/general ledger manajer ke tabel auth_item
        $gl_manager  =  $auth->createPermission('gl-
manager');
        $gl_manager->description  =  'Allow  user  as
General Ledger Manager';
        $auth->add($gl_manager);

        //      menambahkan      akses      sebagai
accounting/general ledger staff ke tabel auth_item
```

```
            $gl_staff   =   $auth->createPermission('gl-
staff');
            $gl_staff->description  =  'Allow  user  as
General Ledger Staff';
            $auth->add($gl_staff);

            // menambahkan akses sebagai human resource
management manajer ke tabel auth_item
            $hrm_manager = $auth->createPermission('hrm-
manager');
            $hrm_manager->description  =  'Allow  user  as
Human and Resource Manager';
            $auth->add($hrm_manager);

            // menambahkan akses sebagai human resource
management staff ke tabel auth_item
            $hrm_staff  =  $auth->createPermission('hrm-
staff');
            $hrm_staff->description  =  'Allow  user  as
Human and Resource Staff';
            $auth->add($hrm_staff);

            // menambahkan akses sebagai inventory and
warehouse management manajer ke tabel auth_item
            $iwm_manager = $auth->createPermission('iwm-
manager');
            $iwm_manager->description  =  'Allow  user  as
Inventory and Warehouse Management Manager';
            $auth->add($iwm_manager);

            // menambahkan akses sebagai inventory and
warehouse management staff ke tabel auth_item
            $iwm_staff  =  $auth->createPermission('iwm-
staff');
            $iwm_staff->description  =  'Allow  user  as
Inventory and Warehouse Management Staff';
            $auth->add($iwm_staff);

            //  menambahkan  akses  sebagai  production
planning manajer ke tabel auth_item
            $pp_manager  =  $auth->createPermission('pp-
manager');
            $pp_manager->description  =  'Allow  user  as
Producton Planning Manager';
            $auth->add($pp_manager);

            //  menambahkan  akses  sebagai  production
planning staff ke tabel auth_item
            $pp_staff   =   $auth->createPermission('pp-
staff');
```

```
                $pp_staff->description  =  'Allow  user  as
        Producton Planning Staff';
                $auth->add($pp_staff);

                //  menambahkan  akses  sebagai  purchasing
        manajer ke tabel auth_item
                $pur_manager = $auth->createPermission('pur-
        manager');
                $pur_manager->description  =  'Allow  user  as
        Purchasing Manager';
                $auth->add($pur_manager);

                // menambahkan akses sebagai purchasing staff
        ke tabel auth_item
                $pur_staff  =  $auth->createPermission('pur-
        staff');
                $pur_staff->description  =  'Allow  user  as
        Purchasing Staff';
                $auth->add($pur_staff);

                //  menambahkan  akses  sebagai  sales  and
        distribution manajer ke tabel auth_item
                $sd_manager  =  $auth->createPermission('sd-
        manager');
                $sd_manager->description  =  'Allow  user  as
        Sales and Distribution Manager';
                $auth->add($sd_manager);

                //  menambahkan  akses  sebagai  sales  and
        distribution staff ke tabel auth_item
                $sd_staff   =   $auth->createPermission('sd-
        staff');
                $sd_staff->description = 'Allow user as Sales
        and Distribution Staff';
                $auth->add($sd_staff);
    }
```

Kode Sumber 4.3.3 Kode Fungsi Tabel AuthItem

```
public function actionItemChild(){
        $auth = Yii::$app->authManager;

        //admin dapat mengakses semua daftar izin akses
        $am_manager = $auth->createPermission('am-manager');
        $am_staff = $auth->createPermission('am-staff');
        $ap_manager = $auth->createPermission('ap-manager');
        $ap_staff = $auth->createPermission('ap-staff');
        $ar_manager = $auth->createPermission('ar-manager');
        $ar_staff = $auth->createPermission('ar-staff');
        $fi_manager = $auth->createPermission('fi-manager');
        $fi_staff = $auth->createPermission('fi-staff');
```

```
        $gl_manager = $auth->createPermission('gl-manager');
        $gl_staff = $auth->createPermission('gl-staff');
        $hrm_manager      =      $auth->createPermission('hrm-
    manager');
        $hrm_staff = $auth->createPermission('hrm-staff');
        $iwm_manager      =      $auth->createPermission('iwm-
    manager');
        $iwm_staff = $auth->createPermission('iwm-staff');
        $pp_manager = $auth->createPermission('pp-manager');
        $pp_staff = $auth->createPermission('pp-staff');
        $pur_manager      =      $auth->createPermission('pur-
    manager');
        $pur_staff = $auth->createPermission('pur-staff');
        $sd_manager = $auth->createPermission('sd-manager');
        $sd_staff = $auth->createPermission('sd-staff');

        $admin = $auth->createRole('admin');
        $auth->add($admin);
        $auth->addChild($admin, $am_manager);
        $auth->addChild($admin, $am_staff);
        $auth->addChild($admin, $ap_manager);
        $auth->addChild($admin, $ap_staff);
        $auth->addChild($admin, $ar_manager);
        $auth->addChild($admin, $ar_staff);
        $auth->addChild($admin, $fi_manager);
        $auth->addChild($admin, $fi_staff);
        $auth->addChild($admin, $gl_manager);
        $auth->addChild($admin, $gl_staff);
        $auth->addChild($admin, $hrm_manager);
        $auth->addChild($admin, $hrm_staff);
        $auth->addChild($admin, $iwm_manager);
        $auth->addChild($admin, $iwm_staff);
        $auth->addChild($admin, $pp_manager);
        $auth->addChild($admin, $pp_staff);
        $auth->addChild($admin, $pur_manager);
        $auth->addChild($admin, $pur_staff);
        $auth->addChild($admin, $sd_manager);
        $auth->addChild($admin, $sd_staff);
             }
```

Kode Sumber 4.3.4 Kode Fungsi Tabel ItemChild

```
public function actionAuthAssignment(){
        $auth = Yii::$app->authManager;

        $admin = $auth->createRole('admin');

        $auth->assign($admin, 1);
    }
```

Kode Sumber 4.3.5 Kode Fungsi Tabel AuthAssignment

4.4 Implementasi Multitenancy

Pada subbab ini akan dibahas mengenai implementasi alur proses aplikasi yang telah dirancang pada Bab III. Alur proses aplikasi akan dibahas mulai dari pengambilan data partisipan, hingga proses peningkatan level dan penghentian pada setiap *training*. Secara rinci mengenai implementasi *RBAC* dijabarkan sebagai berikut:

4.4.1 Membuat Halaman Muka Tenant

Pada implementasi membuat halaman tenant ini dilakukan pembuatan halaman tenant secara sederhana, kemudian ditambahkan pembuatan database untuk setiap tenant. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.4.1.

```
// mendapatkan nilai yang dimasukkan dari view
 $host = "10.151.64.182";
        $tenant = Yii::$app->request->post('tenant');
        $database = Yii::$app->request->post('database');

// membuka koneksi pada untuk memasukkan data tenant
$koneksidb = mysqli_connect($host, "root", "",
        "multitenant");
if ($koneksidb->connect_error) {
    die("Connection failed: " . $koneksidb->connect_error);
                }
        // query untuk memasukkan data tenant
                $insert     =     "insert     into     tenant
        (tenant,database_name,created_at)              values
        ('".$tenant."','".$database."',NOW())";

 if($tenant != "" && $database != ""){
      $koneksidb->query($insert);
 }
// memutuskan koneksi ke server
          $koneksidb->close();
```

Kode Sumber 4.4.1 Pembuatan Halaman Muka Tenant

4.4.2 Menambahkan Database untuk Tenant Baru

Pada tahap ini dilakukan penambahan database untuk tenant baru dengan cara melakukan konfigurasi pada server node mysql. Kode implementasi yang dimaksud ditunjukkan oleh Kode Sumber 4.2.2.

```
 'mysql -u root -p;' lalu tekan enter
```

```
'create database <nama database>;' lalu tekan enter
'use <nama database>;' lalu tekan enter
 e /tmp/mysql-dump/final-db.sql
```

Kode Sumber 4.4.2 Penambahan Database Tenant Baru

4.4.3 Login Tenant

Setelah proses pembuatan database dan replikasi pada tahap 4.4.2 selesai, maka tenant melakukan login dengan memilih salah satu database, nama database yang dipilih tersebut disimpan dalam session dan akan digunakan untuk koneksi yang akan dibuat. Kode implementasi yang dimaksud ditunjukkan oleh kode sumber 4.4.3.

```
$host = '10.151.64.182';  // node mysql
      $port = 3306;
$waitTimeoutInSeconds = 1;
$rc =
       @fsockopen($host,$port,$errCode,$errStr,$waitTimeoutIn
       Seconds);
if(is_resource($rc)){
   $_SESSION['dbserver_ip'] = "10.151.64.182"; // node mysql
   // penyimpanan nama database sebagai session untuk
        multitenancy
   $dbname = isset($_SESSION['database_name']) ?
        $_SESSION['database_name'] : 'test';
} else {
   $_SESSION['dbserver_ip'] = "10.151.64.203";  // node mysql
}
    $connection = [
        'class' => 'yii\db\Connection',
        'dsn' =>
       'mysql:host='.$_SESSION['dbserver_ip'].';dbname='.$dbn
       ame.'',
        'username' => 'root',
        'password' => '',
        'charset' => 'utf8'
    ];
       return $connection;
```

Kode Sumber 4.4.3 Login Tenant

4.5 Implementasi Program pada *Modul Inventory and Warehouse Management*

Pada bagian ini akan dijelaskan mengenai implementasi yang terdapat pada Modul *Inventory and Warehouse Management* yang

terbagi menjadi tampilan halaman utama atau *dashboard* dan beberapa sub modul, antara lain: *Warehouse Data Master (Warehouse, Inventory, Storage, Storage Detail, Bin, Quant), Item Data Master (UoM, Item Type, Item Master, Item Detail, Stock), Good Receipt, Good Issue, Transfer Posting dan Warehouse Movement.* Secara rinci mengenai implementasi lapisan antarmuka Modul *Inventory and Warehouse Management* dijabarkan sebagai berikut:

4.5.1 Halaman Utama Modul *Inventory and Warehouse Management*

Halaman utama modul *Inventory and Warehouse Management* menampilkan informasi pencatatan perpindahan barang masuk dan keluar yang ditampilkan secara grafis. Pengaturan tampilan tersebut diatur di kelas *DefaultController* dalam fungsi *actionIndex()*. Kode tampilan yang dimaksud ditunjukkan oleh kode sumber 4.5.1.

```
public function actionIndex($id = "1")
    {
        $items = IwmItemMaster::find()->all();
        if($id != null){
                $see_item = IwmItemMaster::findOne($id);
        }
        $wm_in = IwmWarehouseMovement::find()
                ->select('date_taken, sum(item_quantity) as
   item_quantity')
                ->where('from_storage_id in (
                        select id
                        from iwm_storage
                        where storage_type = "Receiving"
                        )
                        and to_storage_id not in (
                                select id
                                from iwm_storage
                                where storage_type = "Defect"
                                )
                        and item_id in (
                                select id
                                from iwm_item_detail
                                where item_id = '.$id.'
                                )')
                ->groupBy('date_taken')->asArray()->all();
        $wm_out = IwmWarehouseMovement::find()
```

```
                ->select('date_taken, sum(item_quantity) as
    item_quantity')
                ->where('from_storage_id not in (
                        select id
                        from iwm_storage
                        where storage_type = "Defect"
                        )
                        and to_storage_id in (
                                select id
                                from iwm_storage
                                where storage_type =
    "Shipping" or storage_type = "Production"
                                )
                        and item_id in (
                                select id
                                from iwm_item_detail
                                where item_id = '.$id.'
                                )')
                ->groupBy('date_taken')->asArray()->all();
            return $this->render('index',[
                'items' => $items,
                'see_item' => $see_item,
                'wm_in' => $wm_in,
                'wm_out' => $wm_out,
                ]);
        }
```

Kode Sumber 4.5.1. Halaman Utama Modul *Inventory and Warehouse Management*

4.5.2 **Melihat Daftar *Warehouse***

Pada implementasi melihat daftar *Warehouse*, sistem menampilkan seluruh daftar warehouse yang berhasil ditambahkan pada *warehouse*. Pengaturan tampilan ini diatur pada kelas *IwmWarehouseController* fungsi *actionIndex()* Kode untuk menampilkan seluruh transaksi tersebut ditunjukkan pada kode sumber 4.5.2.

```
public function actionIndex()
    {
        $searchModel = new IwmWarehouseSearch();
        $dataProvider = $searchModel->search(Yii::$app-
>request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
```

```
        ]);
    }
```

Kode Sumber 4.5.2 Melihat Daftar *Warehouse*

4.5.3 Menambah *Warehouse*

Pada implementasi menambah *Warehouse*, sistem menambahkan transaksi baru ke dalam *Warehouse*. Pengaturan manambah transaksi baru ini diatur pada kelas *IwmWarehouseController* fungsi *actionCreate()*. Kode untuk menampilkan seluruh transaksi tersebut ditunjukkan pada kode sumber 4.5.3.

```
public function actionCreate()
    {
        $model = new IwmWarehouse();

        if ($model->loadAll(Yii::$app->request->post()) &&
$model->saveAll()) {
            return $this->redirect(['view', 'id' => $model-
>id]);
        } else {
            return $this->render('create', [
                'model' => $model,
            ]);
        }
    }
```

Kode Sumber 4.5.3 Menambah *Warehouse*

4.5.4 Menyunting *Warehouse*

Pada implementasi menyunting *Warehouse*, sistem memperbarui data transaksi pada *Warehouse*. Pengaturan menyunting transaksi ini diatur pada kelas *IwmWarehouseController* fungsi *actionUpdate()*. Kode untuk menyunting *warehouse* ditunjukkan pada kode sumber 4.5.4.

```
public function actionUpdate($id)
    {
        $model = $this->findModel($id);

        if ($model->loadAll(Yii::$app->request->post()) &&
$model->saveAll()) {
```

```
            return $this->redirect(['view', 'id' => $model-
>id]);
        } else {
            return $this->render('update', [
                'model' => $model,
            ]);
        }
    }
```

Kode Sumber 4.5.4 Menyunting *Warehouse*

4.5.5 Menghapus *Warehouse*

Pada implementasi menghapus *warehouse*, sistem menghapus data transaksi pada *Warehouse*. Pengaturan menyunting transaksi ini diatur pada kelas *IwmWarehouseController* fungsi *actionDelete()*. Kode untuk menghapus *warehouse* ditunjukkan pada kode sumber 4.5.5.

```
public function actionDelete($id)
    {
        $this->findModel($id)->deleteWithRelated();

        return $this->redirect(['index']);
    }
```

Kode Sumber 4.5.5 Menghapus *Warehouse*

4.5.6 Memproses Good Receipt

Pada implementasi memproses daftar *Good Receipt*, sistem akan memproses segala yang berkaitan dengan proses transaksi *Good Receipt*. Pengaturan proses ini diatur pada kelas *IwmGoodReceiptHController* dengan menggunakan salah satu fungsi sesuai dengan data yang terdapat pada receipt itu sendiri, diantaranya actionDefect(), actionNewPurchase(), actionDumpPurchase(), actionUpdatePurchase(), actionnewProduction(), actionDumpProduction(), actionUpdateProduction(). Kode untuk memproses seluruh transaksi tersebut ditunjukkan pada kode sumber 4.5.46.

```
public function actionDefect($id)
{
   …
```

```
    }
    public function actionNewPurchase($id)
    {
        IwmgoodreceipthController::actionDeleteDefect($id);
        IwmgoodreceipthController::actionFromPurchase($id);
        IwmgoodreceipthController::actionCreatePurchase($id);
        return $this->redirect(['index']);
    }
    public function actionDumpPurchase($id){
        IwmgoodreceipthController::actionDeleteDefect($id);
        IwmgoodreceipthController::actionFromPurchase($id);
        return $this->redirect(['index']);
    }
    public function actionUpdatePurchase($id){
        IwmgoodreceipthController::actionFromPurchase($id);
        return $this->redirect(['index']);
    }

    public function actionNewProduction($id)
    {
        IwmgoodreceipthController::actionCreateProduction($id);
        IwmgoodreceipthController::actionDeleteDefect($id);
        IwmgoodreceipthController::actionFromProduction($id);
        return $this->redirect(['index']);
    }
    public function actionDumpProduction($id){
        IwmgoodreceipthController::actionDeleteDefect($id);
        IwmgoodreceipthController::actionFromProduction($id);
        return $this->redirect(['index']);
    }
    public function actionUpdateProduction($id){
        IwmgoodreceipthController::actionFromProduction($id);
        return $this->redirect(['index']);
    }
```

Kode Sumber 4.5.6 Fungsi yang terdapat pada Mengelola Good Receipt

Pada kode sumber 4.5.46, actionDefect() akan melakukan proses lebih lanjut terhadap barang yang tercatat sebagai barang "Defect" atau barang rusak yang terdapat pada salah satu data transaksi pada Good Receipt. actionNewPurchase() adalah fungsi yang akan dijalankan ketika barang rusak akan dikembalikan dan melakukan proses pengembalian barang. actionDumpPurchase() adalah fungsi yang akan dijalankan ketika barang rusak akan dibuang dan tidak melakukan proses pengembalian barang. actionUpdatePurchase() adalah fungsi yang akan dijalankan ketika kerusakan barang tidak terlalu parah dan tetap dimasukkan ke dalam gudang. actionNewProduction() adalah fungsi yang akan dijalankan ketika barang hasil produksi mengalami kerusakan dan harus melakukan produksi ulang. actionDumpProduction() adalah fungsi yang dijalankan ketika barang hasil produksi mengalami kecacatan dan akan dibuang oleh sistem serta tidak melakukan produksi ulang. actionUpdateProduction() adalah fungsi yang dijalankan ketika barang hasil produksi mengalami kecacatan namu tetap dimasukkan ke gudang dan tidak melakukan proses produksi ulang. Untuk lebih jelasnya, dapat dilihat pada lampiran Kode Sumber A.1.

4.5.7 Memproses Good Issue

Pada implementasi memproses daftar *Good Issue*, sistem memproses salah satu data pada *Good Issue* untuk kemudian melakukan seluruh proses bisnis yang terdapat pada Inventory untuk proses ini. Pengaturan proses ini diatur pada kelas *IwmGoodIssueHController*. Kode untuk menampilkan seluruh transaksi tersebut ditunjukkan pada lampiran kode sumber A.2.

4.5.8 Memproses Transfer Posting

Pada implementasi memproses *Transfer Posting*, sistem memproses salah satu transaksi yang terdapat pada halaman utama *Transfer Posting* sesuai dengan proses bisnis yang terdapat pada sistem. Pengaturan proses ini diatur pada kelas

IwmTransferPostingHController. Kode untuk menampilkan seluruh transaksi tersebut ditunjukkan pada lampiran kode sumber A.3.

4.6 Implementasi Antarmuka Pengguna

4.6.1 Antarmuka Melihat Halaman Utama Modul Inventory and Warehouse Management

Pada antarmuka ini pengguna dapat melihat grafik catatan perpindahan barang perhari. Pengguna juga dapat memilih Item yang ingin dilihat catatan perpindahan barangnya. Tampilan antarmuka ini dapat dilihat pada lampiran Gambar A.8.

4.6.2 Antarmuka Melihat Halaman Utama Submodul Pilihan

Pada antarmuka ini pengguna dapat melihat halaman utama yang berisikan daftar-daftar data dari submodul pilihan yang terdapat pada sistem. Pengguna juga dapat memilih melakukan kegiatan menyunting, melihat rincian, dan menghapus salah satu data tertentu serta membuat data baru. Tampilan antarmuka ini dapat dilihat pada lampiran Gambar A.1. Antarmuka ini diimplementasikan pada submodul *Warehouse, Inventory, Storage, Storage Detail, Bin, Quant, Item Type, Item Master, Item Detail, Stock, Good Receipt, Good Issue, Transfer Posting, Warehouse Movement.*

4.6.3 Antarmuka Menambah Data Submodul Pilihan

Pada antarmuka ini pengguna dapat mengisi data yang dibutuhkan terkait submodul yang dipilih dan menekan tombol "*Create*" untuk perintah pembuatan. Tampilan antarmuka ini dapat dilihat pada lampiran Gambar A.2. Antarmuka ini diimplementasikan pada submodul *Warehouse, Inventory, Storage, Storage Detail, Bin, Quant, Item Type, Item Master, Item Detail, Stock, Good Receipt, Good Issue, Transfer Posting, Warehouse Movement.*

4.6.4 **Antarmuka Menyunting Data Submodul Pilihan**

Pada antarmuka ini pengguna dapat merubah data terkait dengan submodul pilihan dan menekan tombol "*Update*" untuk perintah penyimpanan perubahan data. Tampilan antarmuka ini dapat dilihat pada lampiran Gambar A.3. Antarmuka ini diimplementasikan pada submodul *Warehouse, Inventory, Storage, Storage Detail, Bin, Quant, Item Type, Item Master, Item Detail, Stock, Good Receipt, Good Issue, Transfer Posting, Warehouse Movement.*

4.6.5 **Antarmuka Menghapus Data Submodul Pilihan**

Antarmuka ini berapa halaman *pop-up* untuk melakukan konfirmasi penghapusan data dengan menekan tombol "*Yes*" atau pembatalan dengan menekan tombol "*Cancel*". Tampilan antarmuka ini dapat dilihat pada lampiran Gambar A.4. Antarmuka ini diimplementasikan pada submodul *Warehouse, Inventory, Storage, Storage Detail, Bin, Quant, Item Type, Item Master, Item Detail, Stock, Good Receipt, Good Issue, Transfer Posting, Warehouse Movement.*

4.6.6 **Antarmuka Memproses Good Receipt**

Antarmuka ini berapa halaman *pop-up* untuk melakukan konfirmasi pemrosesan data *receipt* jika tidak terdapat barang "Defect" atau barang rusak akan tertampil pop-up konfirmasi dengan menekan tombol "Yes" atau pembatalan dengan menekan tombol "Cancel". Jika terdapat barang rusak maka terdapat 4 pilihan dimana untuk membuang barang rusak dengan menekan tombol "*Dump Item*", membiarkan barang rusak disimpan di gudang dengan menekan tombol "*Ignore Item*", Melakukan pengembalian barang / memproduksi ulang barang dengan menekan tombol "*Return Item / Reproduct Item*" atau pembatalan dengan menekan tombol "*Cancel*". Tampilan antarmuka ini dapat dilihat pada lampiran Gambar A.5.

4.6.7 Antarmuka Memproses Good Issue

Antarmuka ini berapa halaman *pop-up* untuk melakukan konfirmasi pemrosesan data issue dengan menekan tombol "Yes" atau pembatalan dengan menekan tombol "Cancel". Tampilan antarmuka ini dapat dilihat pada lampiran Gambar A.6.

4.6.8 Antarmuka Memproses Transfer Posting

Antarmuka ini berapa halaman *pop-up* untuk melakukan konfirmasi pemrosesan data transfer dengan menekan tombol "Yes" atau pembatalan dengan menekan tombol "Cancel". Tampilan antarmuka ini dapat dilihat pada lampiran Gambar A.7.

[Halaman ini segaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas hasil dan pembahasan pada aplikasi yang dikembangkan. Pada bab ini akan dijelaskan tentang data yang digunakan, hasil yang didapatkan dari penggunaan perangkat lunak dan uji coba yang dilakukan pada perangkat lunak yang telah dikerjakan untuk menguji apakah fungsionalitas aplikasi telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya.

5.1 Lingkungan Uji Coba

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji implementasi pembuatan sistem pada tugas akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

1. Perangkat keras
   a. Prosesor: Intel® Core™ i5 CPU @ 2.10GHz
   b. Memori (RAM): 4 GB
   c. Tipe sistem: 32-bit sistem operasi
2. Perangkat lunak
   a. Sistem operasi: Windows 7 Professional
   b. Perangkat pengembang: Python-2.7.11

5.2 Skenario Pengujian

Dalam uji coba yang dilakukan dalam tugas akhir ini memiliki beberapa tahapan yang dijelaskan pada subbab ini.

5.2.1 Business Plan

Business plan yang dibuat adalah sebuah perusahaan manufaktur sepeda. Perusahaan manufaktur dipilih karena memiliki tingkat kompleksitas proses bisnis yang paling tinggi. Sehingga aplikasi ERP yang dikembangkan dapat mengakomodasi proses bisnis berbagai jenis perusahaan.

Berikut adalah *business plan* yang telah dibuat:

1. *Finished Goods*

Tabel 3.1. *Finished Goods* Data

No	Nama Barang
1	Red Deluxe Touring Bike
2	Red Profesional Touring Bike

Perusahaan ini memproduksi dua jenis sepeda. Dua jenis sepeda tersebut memiliki bahan dasar yang berbeda, Deluxe Touring Bike berbahan dasar aluminium, sedangkan Profesional Touring Bike berbahan dasar karbon.

2. *Assets*

Tabel 3.2. *Assets Data*

No.	**Name**	**Quantity**
1	Tanah	180x150 m2
2	Kantor	50x50 m2
3	Parkir	60x50 m2
4	Kantin	40x60 m2
5	Pabrik	108x80 m2
6	Pengolahan Limbah	20x40 m2
7	Raw Materials Inventory	40x60 m2
8	Semi-Finished Good Inventory	40x60 m2
9	Finished Good Inventory	40x60 m2
10	Welding Machine	2 line
11	Molding Machine	2 line
12	Laser Cutting Machine	2 line
13	Spray Painting Machine	2 line
14	Testing Machine	2 line
15	Truck	10
16	Forklift	6
17	Heavy Forklift	6

Perusahaan ini memiliki 17 aset pada perencanaannya. Dijabarkan pada tabel 3.2. berikut dengan kuantitas dari masing-masing aset yang ada.

3. *Raw Materials*

Tabel 3.3. *Raw Materials* Data

No.	Raw Materials Name	Weight	UoM
1	Tire	1.2	Pcs
2	Seat Kit	0.5	Pcs
3	Chain	1.5	Pcs
4	Gear	1.8	Pcs
5	Brake	0.6	Pcs
6	Handle Bar	1.2	Pcs
7	Pedal	0.4	Pcs
8	Aluminium	2	m
9	Carbon Fiber	2	m
10	Red Paint 20KG	20	Big Drum
11	Velg	2.3	Pcs
12	Tube	0.4	Pcs
13	Hex Nut 5mm	0.04	Pcs
14	Lock Washer 5mm	0.06	Pcs
15	Socket Head Bolt 5mm	0.03	Pcs

Perencanaan yang diadakan oleh perusahaan ini akan menggunakan 15 macam *raw materials* seperti yang dijabarkan pada tabel 3.3. 15 *raw materials* ini ditentukan dari *bill of material* yang akan diterangkan di tabel 3.4.

- *Bill of Materials*

Tabel 3.4. Bill of Materials Data

No.	Materials Needed	Quantity	UoM	Materials Produced
1	Aluminium	5	M	Frame for Deluxe Touring Bike with Red Color
2	Paint drum with Red Color 20KG	0.05	Big drum	
3	Carbon Fiber	5	M	Frame for Professional Bike with Red Color
4	Paint drum with Red Color 20KG	0.05	Big drum	
5	Hex Nut 5 mm	2	Pcs	Wheel's Bike
6	Lock Washer 5 mm	2	Pcs	

No.	Materials Needed	Quantity	UoM	Materials Produced
7	Socket Head Bolt 5mm	1	Pcs	
8	Tire's Bike	1	Pcs	
9	Wheel's Tube	1	Pcs	
10	Wheel's Velg	1	Pcs	
11	Frame for Deluxe Touring Bike with Red Color	1	Pcs	Deluxe Touring Bike with Red Main Color
12	Wheel's Bike	2	Pcs	
13	Chain's Bike	1	Pcs	
14	Handle Bar's Bike	1	Pcs	
15	Seat Kit's Bike	1	Pcs	
16	Pedal's Bike	2	Pcs	
17	Gear's Bike	1	Pcs	
18	Front and Rear Brake's Bike	1	Pcs	
19	Frame for Deluxe Touring Bike with Red Color	1	Pcs	Professional Bike with Red Color
20	Wheel's Bike	2	Pcs	
21	Chain's Bike	1	Pcs	
22	Handle Bar's Bike	1	Pcs	
23	Seat Kit's Bike	1	Pcs	
24	Pedal's Bike	2	Pcs	
25	Gear's Bike	1	Pcs	
26	Front and Rear Brake's Bike	1	Pcs	

Bill of Materials yang dijabarkan pada tabel 3.4. menjelaskan material-material yang dibutuhkan untuk memproduksi suatu barang yang dibutuhkan pada perusahaan ini.

4. *Resources*

Jumlah pegawai:

- *Direct labour* : 150 orang
- *Indirect labour* : 32 orang

Mesin (2 line):

- *Welding Machine*
- *Molding Machine*
- *Laser Cutting Machine*
- *Spray Painting Machine*
- *Testing Machine*

5. Perencanaan Keuangan

Total pembelian aset aktif	: Rp 27.997.311.200,00
Total pembelian *raw materials*	: Rp 88.615.744.062,50
Biaya lain-lain	: Rp 3,386,944,737.50 +
Peminjaman dari Bank	: Rp 120.000.000.000,00

5.2.2 Prosedur Simulasi Pasar

I. Peserta

5 perusahaan (45 orang bisa lebih) yang terdiri dari Perusahaan A, Perusahaan B, Perusahaan C, Perusahaan D dan Perusahaan E.

II. Prosedur Simulasi

a. Make to Stock

1. Perusahaan menentukan sendiri supplier dari bahan baku yang akan dibeli berdasarkan data per supplier (histori defect rate, harga barang, toleransi pembayaran, denda dan diskon) yang diberikan pasar.
2. Perusahaan melakukan produksi berdasarkan hasil forecasting dari histori serapan.
3. Perusahaan menentukan sendiri strategi penyebaran finished goods nya pada pasar yang tersedia (Grosir, Midi, Retail).
4. Perusahaan menyiapkan COGS dan biaya iklan yang diberikan kepada pasar yang akan digunakan pasar untuk menentukan serapan.

5. Perusahaan menjual finished goods nya sesuai serapan yang diberikan oleh pasar.

b. Make to Order

1. Perusahaan menentukan sendiri supplier dari bahan baku yang akan dibeli berdasarkan data per supplier (histori defect rate, harga barang, toleransi pembayaran, denda dan diskon) yang diberikan pasar.
2. Perusahaan menyiapkan COGS dan biaya iklan yang diberikan kepada pasar yang akan digunakan pasar untuk menentukan serapan.
3. Perusahaan melakukan produksi (apabila diperlukan) hingga finished goods sesuai dengan serapan yang diberikan oleh pasar.
4. Perusahaan menjual finished goods nya sesuai serapan yang diberikan oleh pasar.

III. Permodalan

i. Uang Rp. 30.000.000.000,-
ii. 2 mesin testing Rp. 17.179.500,-
iii. 2 mesin laser Rp. 1.083.630.000,-
iv. 2 mesin pengelasan Rp. 396.450.000,-
v. 2 mesin pengecatan Rp. 132.150.000,-
vi. 2 mesin molding Rp. 317.160.000,-
vii. Tanah Rp. 10.000.000.000,-
viii. 3 Gudang Rp. 1.600.000.000,-

IV. Produk

i. Bike Deluxe (Green, Red, Black)
ii. Bike Profesional (Green, Red, Black)

V. Prosedur Simulasi

a. Make to Stock (MTS)

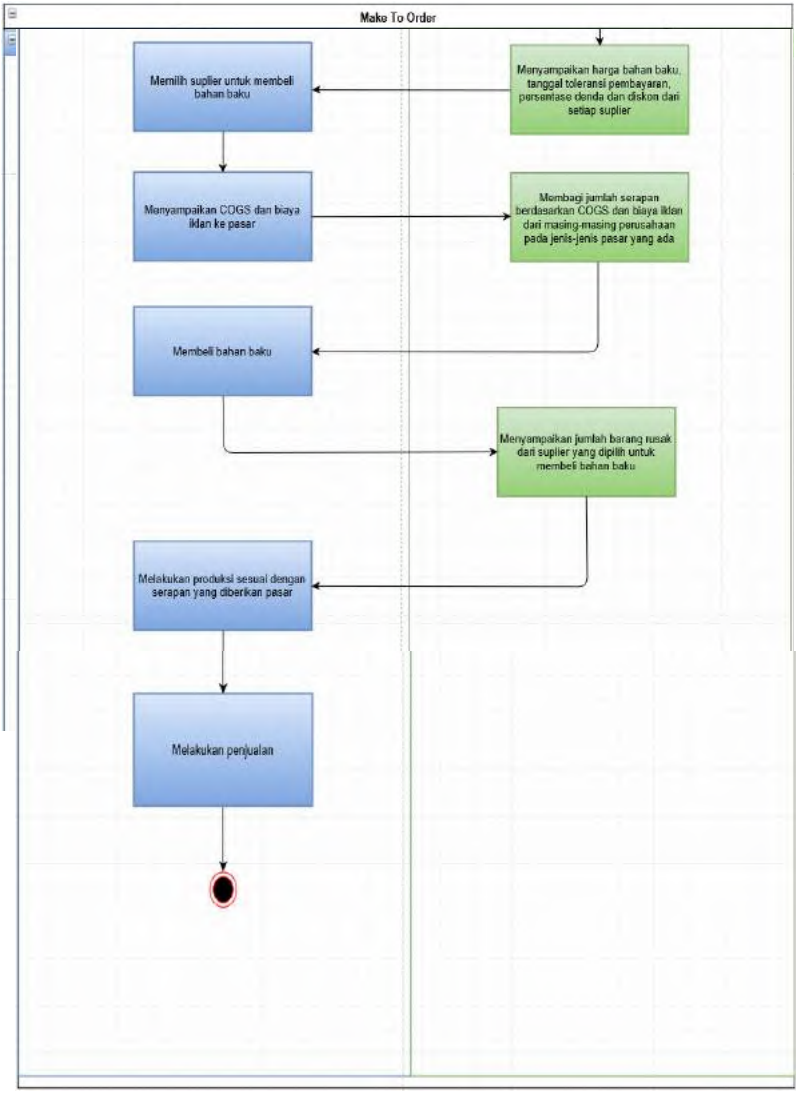
1. Pasar mengeluarkan harga bahan baku, tanggal toleransi pembayaran, persentase denda dan diskon dari setiap supplier.

2. Perusahaan membeli bahan baku dari supplier yang dipilihnya.
3. Pasar menyampaikan jumlah barang rusak dari supplier yang dipilih untuk membeli bahan baku.
4. Perusahaan melakukan produksi berdasarkan hasil forecasting dari histori serapan.
5. Perusahaan memberikan COGS dan biaya iklan kepada pasar yang akan digunakan pasar untuk menentukan serapan.
6. Perusahaan menjual finished goods nya sesuai serapan yang diberikan oleh pasar.

b. Make to Order (MTO)

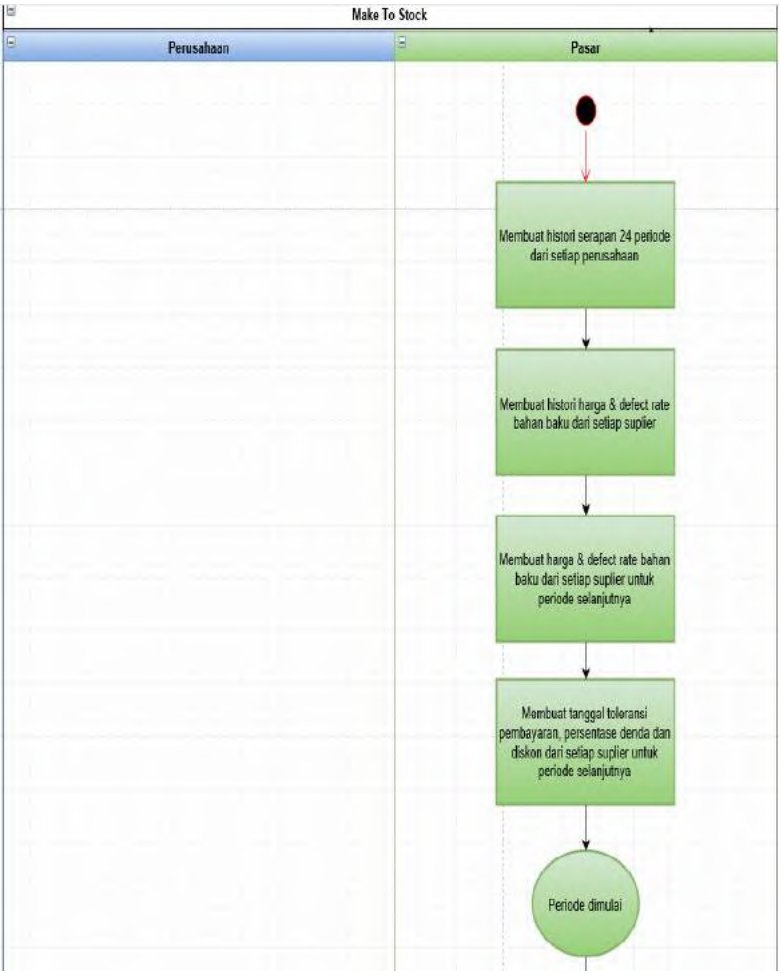
1. Pasar mengeluarkan harga bahan baku, tanggal toleransi pembayaran, persentase denda dan diskon dari setiap supplier.
2. Perusahaan memberikan COGS dan biaya iklan kepada pasar yang akan digunakan pasar untuk menentukan serapan.
3. Perusahaan membeli bahan baku (apabila diperlukan) dari supplier yang dipilihnya.
4. Pasar menyampaikan jumlah barang rusak dari supplier yang dipilih untuk membeli bahan baku.
5. Perusahaan melakukan produksi (apabila diperlukan) hingga finished goods sesuai dengan serapan yang diberikan oleh pasar.
6. Perusahaan menjual finished goods nya sesuai serapan yang diberikan oleh pasar.

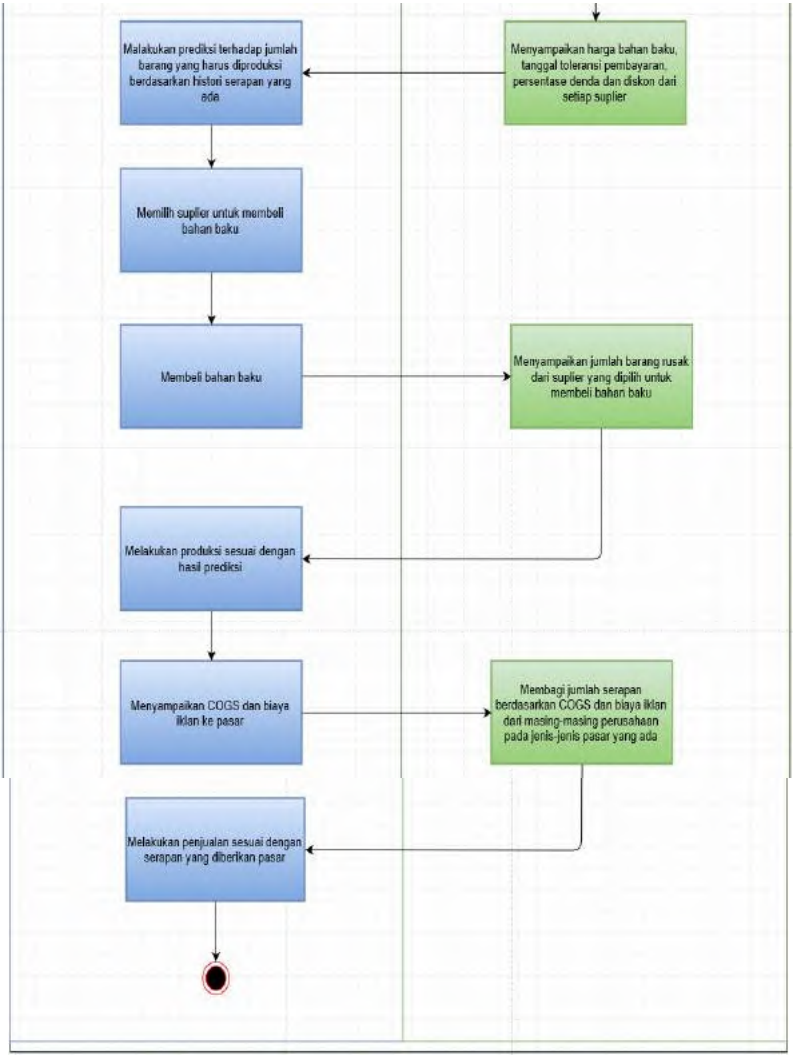
VI. Alur Diagram Make to Order (MTO)



Gambar 5.1 Alur Diagram *Make To Order*

VII. Alur Diagram Make to Stock (MTS)





Gambar 5.2 Alur Diagram Make to Stock

VIII. Aturan Penyerapan Pasar

a. Rumus random banyaknya penyerapan per hari per produk

Hi = Ho +((randbetween(4,6)/10)SD)
atau
Hi = Ho -((randbetween(2,4)/10)SD)
Hi : Jumlah penyerapan hari ini
Ho : Jumlah penyerapan kemarin
SD : standar deviasi

b. Rumus pembagian penyerapan untuk tiap perusahaan per jenis pasar (menggunakan Algoritma Prioritas Wiegers)
Vpj = Mc BMcHj BHj
Vpj : Value perusahaan untuk produk p (Deluxe, Profesional) pada jenis pasar j (Grosir, Medium, Retail)
Mpj : Marketing cost produk p pada jenis pasar j
BMj : Bobot marketing cost pada jenis pasar j
Hpj : COGS produk p pada jenis pasar j

Jenis Pasar	*BHj*	BMj
Grosir	3	1
Medium	2	2
Retail	1	3

$Pij = Tpj \times Vipji=0nVipj$

Pij : Jumlah produk (Deluxe, Profesional) yang diserap oleh perusahaan i pada jenis pasar j (Grosir, Medium, Retail)
Tpj : Total serapan produk p pada jenis pasar j
Vipj: Value perusahaan i untuk produk p pada jenis pasar j
i=0nVipj: Total value dari n perusahaan untuk produk p pada jenis pasar j.

5.2.3 Pengujian Fitur Basis Data Terdistribusi

Pengujian Basis Data Terdistribusi (BDT) dilakukan untuk menguji 2 aspek utama dalam konsep BDT, yaitu replikasi dan *high availability*.

Sistem BDT menggunakan 4 server fisik. Dengan rincian sebagai berikut:

1. *Server* aplikasi, adalah server yang di dalamnya terdapat *web server* beserta file aplikasi ERP.
2. *Management Node*, adalah *server* yang berfungsi sebagai pusat pengaturan sistem BDT.
3. *Data Node*, adalah 2 buah server penyimpan data.

5.2.3.1 Replikasi

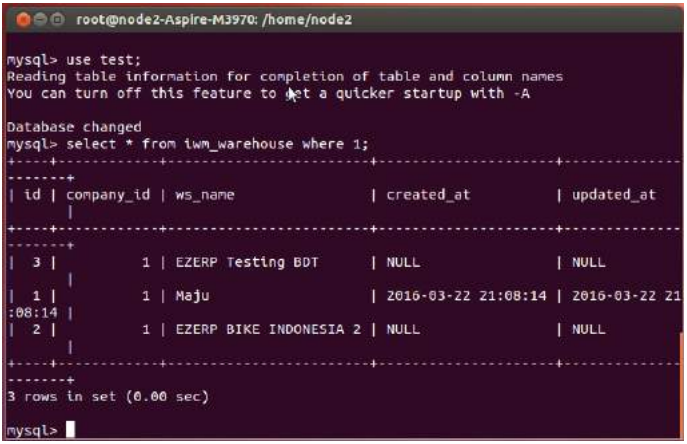
Replikasi adalah penyalinan *table-table* pada basis data ke beberapa *node* fisik yang tersebar. Operasi basis data yang terjadi pada sistem akan didistribusikan ke seluruh *node data* yang terlibat. Sehingga setiap *node data* memiliki struktur *table* dan data yang identik. Berikut langkah-langkah pengujian mekanisme replikasi :

1. Memastikan seluruh *server* dan sistem BDT dalam keadaan *online*.
2. Melakukan penambahan. perubahan, dan penghapusan data melalui aplikasi ERP yang ditunjukkan pada Gambar 5.29.

Gambar 5.3 Pengujian fitur Replikasi pada sistem

3. Melakukan pengecekan terhadap hasil penambahan, perubahan, dan penghapusan data pada seluruh *data node* yang ditampilkan pada Gambar 5.30 dan Gambar 5.31.

Gambar 5.4 Pengujian Fitur Replikasi pada Database Server 1



Gambar 5.5 Pengujian Fitur Replikasi pada Database Server 2

5.2.3.2 ***High Availability***

Aspek lain dari BDT adalah *high availability* adalah kemampuan sistem basis data untuk tetap berjalan normal pada kondisi sebuah *node data* tidak aktif. Operasi basis data yang terjadi akan dieksekusi pada *node data* yang aktif. Di lain sisi, *management node* akan menciptakan *log* operasi-operasi yang terjadi. Sehingga operasi-

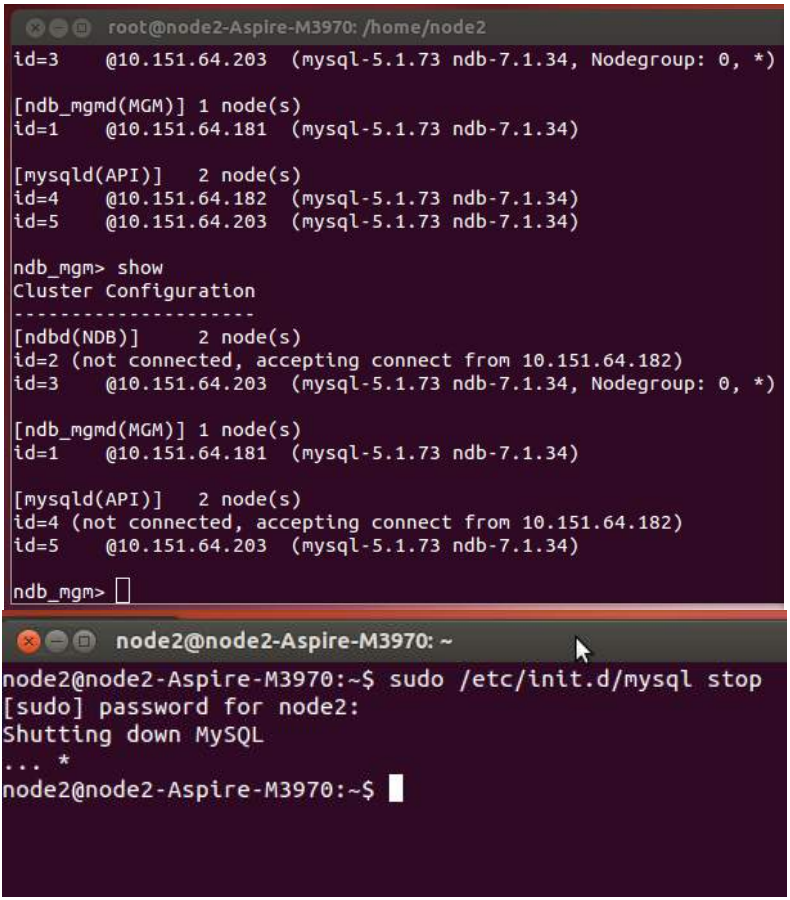
operasi basis data dapat dieksekusi pada *node data* yang kembali aktif. Berikut langkah-langkah pengujian aspek *high availability* :

1. Mematikan *server* aplikasi, *management server* dan sistem BDT pada salah satu *node.* Ditampilkan pada Gambar 5.32.

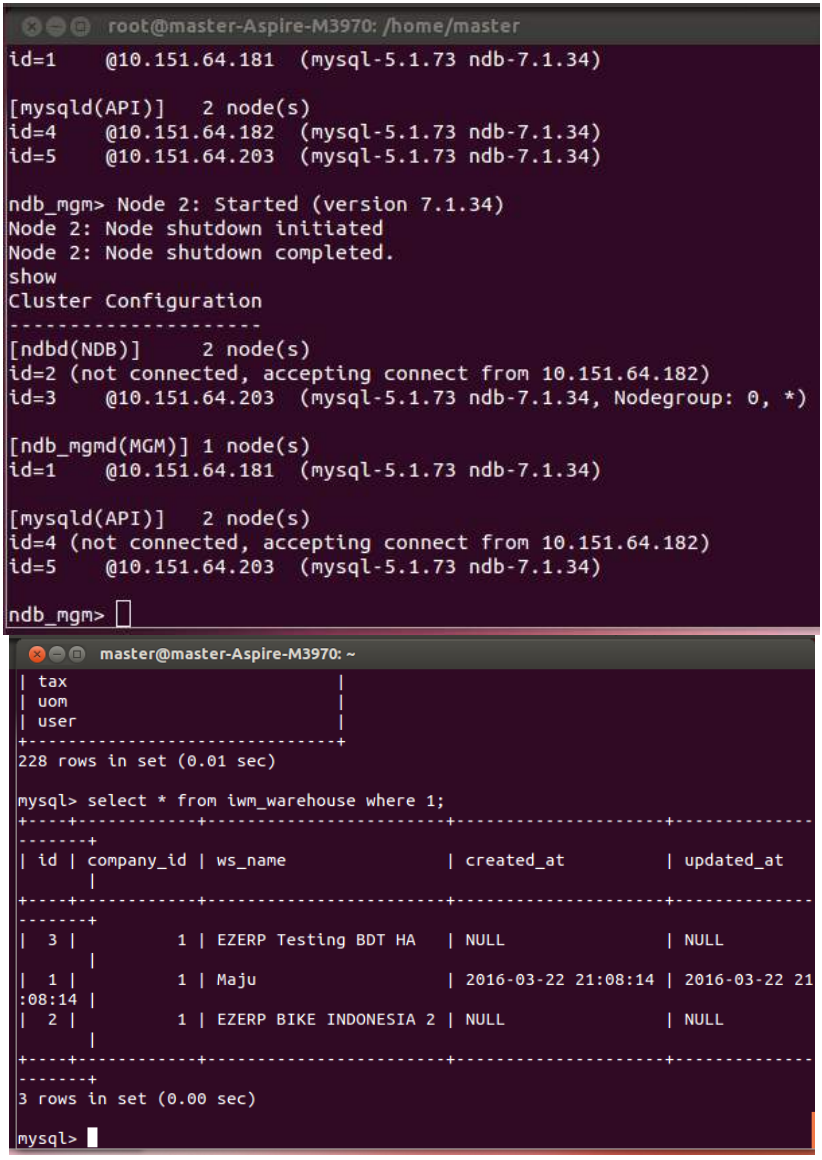
2. Melakukan penambahan, perubahan, dan penghapusan data melalui aplikasi yang ditunjukkan pada Gambar 5.33.

Gambar 5.6 Pengujian Fitur High-Availability pada Sistem

3. Melakukan pengecekan terhadap hasil pemrosesan data pada *node* yang masih aktif yang ditampilkan pada Gambar 5.34.



Gambar 5.7 Pengujian Fitur High-Availability pada Database Server 2



Gambar 5.8 Pengujian Fitur High-Availability pada Database Server 1

5.2.4 **Pengujian RBAC**

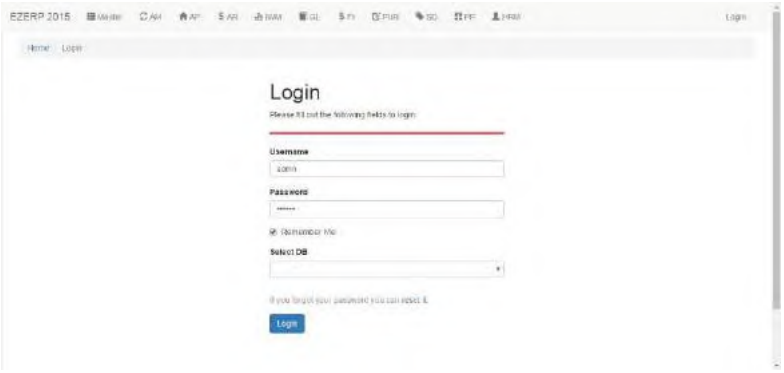
Pengujian ini terdiri dari pengujian mengela *Role Based Access Control (RBAC)*. Rincian skenario pengujian pada kasus penggunaan pengujian mengelola *Role Based Access Control (RBAC)* dapat dilihat pada Tabel 6.11.

Tabel 5.1 Pengujian fitur mengelola Role Based Access Control (RBAC)

ID	UJ.UC.RBAC
Nama	Pengujian mengelola *Role Based Access Control (RBAC).*
Tujuan Pengujian	Menguji kemampuan aplikasi mengelola data *Role Based Access Control (RBAC).*
Skenario 1	***Pengguna membuat user baru.***
Kondisi Awal	Pengguna berada pada halaman dashboard admin.
Data Uji	Inputan data *user* baru:
Langkah Pengujian	Pengguna memilih kegiatan *Add New User*, lalu memasukkan data inputan dan menekan tombol "*Add User*".
Hasil Yang Diharapkan	*User* yang baru terdapat pada halaman daftar *user*.
Hasil Yang Didapat	*User* yang baru terdapat pada halaman daftar *user*.
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman halaman daftar *user* dengan tambahan data *user* yang baru.
Skenario 2	***Pengguna menyunting data user tertentu.***
Kondisi Awal	Pengguna berada pada halaman daftar *user*.
Data Uji	Perubahan data *user*:
Langkah Pengujian	Pengguna memilih kegiatan menyunting *user* tertentu, lalu melakukan perubahan data dan menekan tombol "*Update*".
Hasil Yang Diharapkan	*User* yang disunting mengalami perubahan data sesuai inputan.

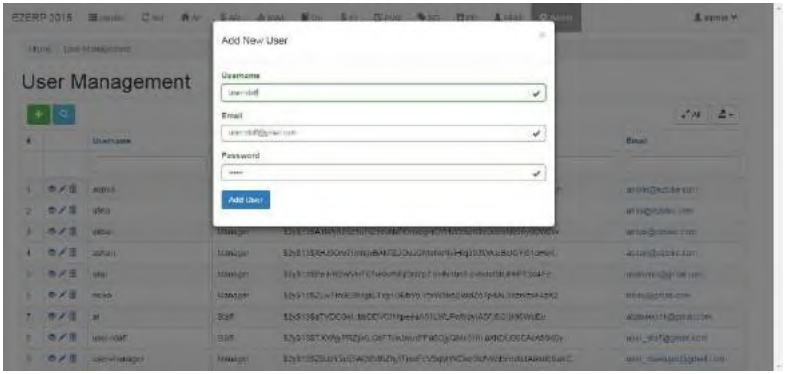
Hasil Yang Didapat	*User* yang disunting mengalami perubahan data sesuai inputan.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman daftar user.
Skenario 3	***Pengguna menghapus user tertentu.***
Kondisi Awal	Pengguna berada pada halaman daftar user.
Data Uji	-
Langkah Pengujian	Pengguna memilih kegiatan menghapus kegiatan tertentu, lalu menekan tombol "*Yes*" untuk konfirmasi penghapusan.
Hasil Yang Diharapkan	*User* yang baru dihapus tidak tampil pada daftar *user*.
Hasil Yang Didapat	*User* yang baru dihapus tidak tampil pada daftar *user*.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman daftar *user*.

Gambar 5.1 menunjukkan proses login oleh admin. Gambar 5.2 menunjukkan hasil setelah proses login oleh admin. Gambar 5.3 menunjukkan proses penginputan data *user* untuk skenario 1. Sedangkan untuk Gambar 5.5 menunjukkan hasil pengujiannya, dimana dapat dilihat penambahan pada halaman daftar user yang baru. Kemudian pada Gambar 5.4 menunjukkan proses penyuntingan data *user* untuk skenario 2. Sedangkan pada Gambar 5.5 dapat dilihat proses penghapusan *user* untuk skenario 3. Dengan melihat hasil pengujian pada ketiga skenario diatas, bisa disimpulkan bahwa Kasus Penggunaan UC.RBAC telah bekerja dengan baik seperti yang diharapkan.

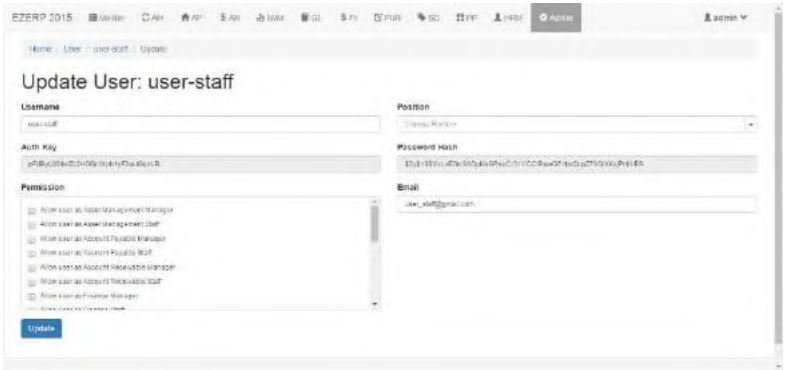


Gambar 5.9 Proses login admin

Gambar 5.10 Tampilan awal setelah login berhasil dilakukan



Gambar 5.11 Proses admin menambahkan user baru



Gambar 5.12 Proses menyunting data user oleh admin

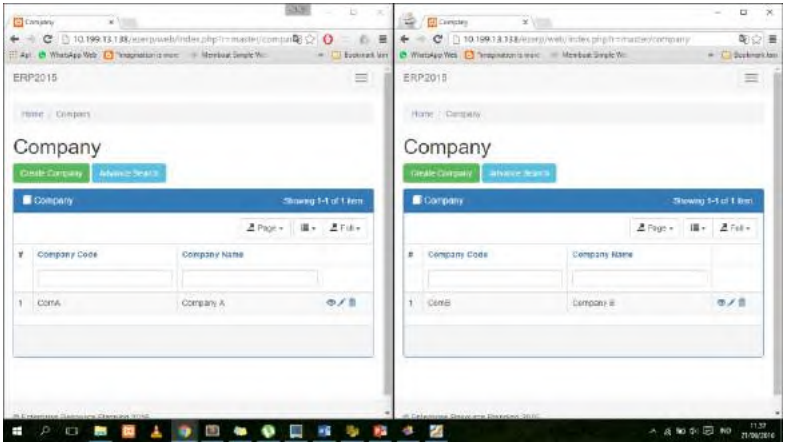
Gambar 5.13 Proses penghapusan user oleh admin

5.2.5 Pengujian Multitenancy

Pengujian ini terdiri dari pengujian mengelola *multitenancy*. Pada pengujian *multitenancy*, *tenant* baru yang akan mendaftarkan *tenant*-nya untuk dapat mengakses sistem memasukkan data berupa nama basis data yang akan dibuat. Apabila basis data berhasil dibuat, maka *tenant* dapat mulai mengakses dengan pengguna awal sebagai administrator. Rincian skenario pengujian pada fitur *multitenancy* dapat dilihat pada Tabel 5.1.

Figure 5.1 Pengujian Fitur *Multitenancy*

ID	**UJ.MTC**
Referensi	*Multitenancy*
Nama	Mengelola *Multitenancy*
Tujuan Pengujian	Menguji kemampuan sistem dalam mengelola *Multitenancy*
Skenario 1	**Pengguna mengakses Tenant**
Kondisi Awal	Pengguna berada pada halaman *login*
Data Uji	Data *Tenant*
Langkah Pengujian	Pengguna memasukkan data uji ke form yang tersedia dan memilih tombol login
Hasil Yang Diharapkan	Pengguna berada di halaman utama dengan data *tenant*-nya
Hasil Yang Didapat	Pengguna berada di halaman utama dengan data *tenant*-nya
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada halaman utama



Gambar 5.14 Hasil Pengujian *Multitenancy*

5.2.6 Pengujian Fungsionalitas

5.2.6.1 Pengujian Fitur Mengelola Item Master

Pengujian ini terdiri dari pengujian mengelola data *Item Master*. Pengujian mengelola data *Item Master* yaitu menambah, menyunting, dan menghapus data *Item Master* yang terdiri dari submodul *Item Type*, *Item Master*, *Item Detail*, *Stock* dimana sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.2.

Tabel 5.2 Tabel Pengujian Fitur Mengelola Item Master

ID	UJ-UC-001
Referensi Kasus Penggunaan	UC-001
Nama	Pengujian Fitur Mengelola Item Master
Tujuan Pengujian	Menguji kemampuan aplikasi mengelola data Item Master
Skenario 1	***Pengguna melihat data Item Type***
Kondisi Awal	Aktor berada pada halaman utama Item Type IWM
Data Uji	Inputan data Item Type :

Langkah Pengujian	Aktor masuk ke halaman membuat data Item Type baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 2	***Aktor menyunting data Item Type***
Kondisi Awal	Aktor berada pada halaman utama Item Type IWM
Data Uji	Perubahan data pada Item Type :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Item Type yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 3	***Aktor menghapus data Item Type***
Kondisi Awal	Aktor berada pada halaman utama Item Type IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Item Type yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Item Type.
Skenario 4	***Pengguna melihat data Item Master***

Kondisi Awal	Aktor berada pada halaman utama Item Master IWM
Data Uji	Inputan data Item Master :
Langkah Pengujian	Aktor masuk ke halaman membuat data Item Master baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 5	***Aktor menyunting data Item Master***
Kondisi Awal	Aktor berada pada halaman utama Item Master IWM
Data Uji	Perubahan data pada Item Master :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Item Master yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 6	***Aktor menghapus data Item Master***
Kondisi Awal	Aktor berada pada halaman utama Item Master IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Item Master yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.

Kondisi Akhir	Pengguna berada pada halaman utama Item Master.
Skenario 7	***Pengguna melihat data Item Detail***
Kondisi Awal	Aktor berada pada halaman utama Item Detail IWM
Data Uji	Inputan data Item Detail :
Langkah Pengujian	Aktor masuk ke halaman membuat data Item Detail baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 8	***Aktor menyunting data Item Detail***
Kondisi Awal	Aktor berada pada halaman utama Item Detail IWM
Data Uji	Perubahan data pada Item Detail :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Item Detail yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 9	***Aktor menghapus data Item Detail***
Kondisi Awal	Aktor berada pada halaman utama Item Detail IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Item Detail yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data

Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Item Detail.
Skenario 10	***Pengguna melihat data Stock***
Kondisi Awal	Aktor berada pada halaman utama Stock IWM
Data Uji	Inputan data Stock :
Langkah Pengujian	Aktor masuk ke halaman membuat data Stock baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 11	***Aktor menyunting data Stock***
Kondisi Awal	Aktor berada pada halaman utama Stock IWM
Data Uji	Perubahan data pada Stock :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Stock yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 12	***Aktor menghapus data Stock***
Kondisi Awal	Aktor berada pada halaman utama Stock IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Stock yang ada pada sistem, lalu melakukan konfirmasi

	penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Stock.

5.2.6.2 Pengujian Fitur Mengelola Warehouse Master

Pengujian ini terdiri dari pengujian mengelola data *Warehouse Master*. Pengujian mengelola data *Warehouse Master* yaitu menambah, menyunting, dan menghapus data *Warehouse Master* yang terdiri dari submodul *Warehouse, Inventory, Storage, Storage Detail, Bin, Quant* dimana sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.3.

Tabel 5.3 Tabel Pengujian Fitur Mengelola Warehouse Master

ID	UJ-UC-002
Referensi Kasus Penggunaan	UC-002
Nama	Pengujian Fitur Mengelola Warehouse Master
Tujuan Pengujian	Menguji kemampuan aplikasi mengelola data Warehouse Master
Skenario 1	***Pengguna melihat data Warehouse***
Kondisi Awal	Aktor berada pada halaman utama Warehouse IWM
Data Uji	Inputan data Warehouse :
Langkah Pengujian	Aktor masuk ke halaman membuat data Warehouse baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 2	***Aktor menyunting data Warehouse***

Kondisi Awal	Aktor berada pada halaman utama Warehouse IWM
Data Uji	Perubahan data pada Warehouse :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Warehouse yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 3	***Aktor menghapus data Warehouse***
Kondisi Awal	Aktor berada pada halaman utama Warehouse IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Warehouse yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Warehouse.
Skenario 4	***Pengguna melihat data Inventory***
Kondisi Awal	Aktor berada pada halaman utama Inventory IWM
Data Uji	Inputan data Inventory :
Langkah Pengujian	Aktor masuk ke halaman membuat data Inventory baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil

Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 5	***Aktor menyunting data Inventory***
Kondisi Awal	Aktor berada pada halaman utama Inventory IWM
Data Uji	Perubahan data pada Inventory :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Inventory yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 6	***Aktor menghapus data Inventory***
Kondisi Awal	Aktor berada pada halaman utama Inventory IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Inventory yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Inventory.
Skenario 7	***Pengguna melihat data Storage***
Kondisi Awal	Aktor berada pada halaman utama Storage IWM
Data Uji	Inputan data Storage :
Langkah Pengujian	Aktor masuk ke halaman membuat data Storage baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.

Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 8	***Aktor menyunting data Storage***
Kondisi Awal	Aktor berada pada halaman utama Storage IWM
Data Uji	Perubahan data pada Storage :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Storage yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 9	***Aktor menghapus data Storage***
Kondisi Awal	Aktor berada pada halaman utama Storage IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Storage yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Storage.
Skenario 10	***Pengguna melihat data Storage Detail***
Kondisi Awal	Aktor berada pada halaman utama Storage Detail IWM
Data Uji	Inputan data Storage Detail :

Langkah Pengujian	Aktor masuk ke halaman membuat data Storage Detail baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 11	***Aktor menyunting data Storage Detail***
Kondisi Awal	Aktor berada pada halaman utama Storage Detail IWM
Data Uji	Perubahan data pada Storage Detail :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Storage Detail yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 12	***Aktor menghapus data Storage Detail***
Kondisi Awal	Aktor berada pada halaman utama Storage Detail IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Storage Detail yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Storage Detail.
Skenario 13	***Pengguna melihat data Bin***

Kondisi Awal	Aktor berada pada halaman utama Bin IWM
Data Uji	Inputan data Bin :
Langkah Pengujian	Aktor masuk ke halaman membuat data Bin baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 14	***Aktor menyunting data Bin***
Kondisi Awal	Aktor berada pada halaman utama Bin IWM
Data Uji	Perubahan data pada Bin :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Bin yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 15	***Aktor menghapus data Bin***
Kondisi Awal	Aktor berada pada halaman utama Bin IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Bin yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.

Kondisi Akhir	Pengguna berada pada halaman utama Bin.
Skenario 16	***Pengguna melihat data Quant***
Kondisi Awal	Aktor berada pada halaman utama Quant IWM
Data Uji	Inputan data Quant :
Langkah Pengujian	Aktor masuk ke halaman membuat data Quant baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 17	***Aktor menyunting data Quant***
Kondisi Awal	Aktor berada pada halaman utama Quant IWM
Data Uji	Perubahan data pada Quant :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Quant yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 18	***Aktor menghapus data Quant***
Kondisi Awal	Aktor berada pada halaman utama Quant IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Quant yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data

Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Quant.

5.2.6.3 Pengujian Fitur Mengelola Barang Datang

Pengujian ini terdiri dari pengujian mengelola data barng datang. Pengujian mengelola data barang datang yaitu menyunting, menghapus dan memproses data barang datang yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.4.

Tabel 5.4 Tabel Pengujian Fitur Mengelola Barang Datang

ID	UJ-UC-003
Referensi Kasus Penggunaan	UC-003
Nama	Pengujian Fitur Mengelola Barang Datang
Tujuan Pengujian	Menguji kemampuan aplikasi mengelola barang datang
Skenario 1	***Aktor menyunting data Good Receipt***
Kondisi Awal	Aktor berada pada halaman utama Good Receipt IWM
Data Uji	Perubahan data pada Good Receipt :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Good Receipt yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 2	***Aktor menghapus data Good Receipt***
Kondisi Awal	Aktor berada pada halaman utama Good Receipt IWM
Data Uji	-

Langkah Pengujian	Pengguna memilih menghapus salah satu data Good Receipt yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Good Receipt.
Skenario 3	***Aktor memproses data Good Receipt***
Kondisi Awal	Aktor berada pada halaman utama Good Receipt IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih memproses salah satu data yang terdapat pada Good Receipt dengan menekan tombol fromPurchase/ fromProduction/ ReturnItem. Akan muncul halaman pemrosesan barang rusak jika pada data yang di proses memiliki data barang rusak di dalamnya dimana kemudian Aktor memilih salah satu pilihan yang akan memproses barang rusak lebih lanjut. Jika pada data yang di proses tidak terdapat barang rusak, maka akan muncul konfirmasi untuk penyelesaian data dimana Aktor akan mengkonfirmasi pemrosesan data.
Hasil Yang Diharapkan	Data yang di proses akan merubah data di sub modul lainnya yang berpengaruh serta status pada data terserbut akan berubah
Hasil Yang Didapat	Data berhasil di proses.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Good Receipt.

Pada skenario 3 yang dijelaskan pada lampiran Gambar A.5 menampilkan proses melihat halaman utama pada submodul *Good Receipt*. Skenario dimulai ketika aktor akan melakukan proses lebih lanjut dengan salah satu data *Good Receipt* dimana aktor akan menekan salah satu tombol from Purchase / from Production / Return Item yang ada di setiap data dapat dilihat pada Lampiran Gambar A.5. Sistem akan menampilkan catatan barang rusak jika terdapat catatan barang rusak

pada data yang akan diproses lebih lanjut. Aktor akan memilih salah satu pilihan yang disediakan dan sistem akan memprosesnya lebih lanjut. Skenario akan berakhir ketika tombol tidak dapat ditekan lagi dan status data tersebut berubah menjadi "*Published*".

5.2.6.4 Pengujian Fitur Mengelola Permintaan Barang

Pengujian ini terdiri dari pengujian mengelola data permintaan barang. Pengujian mengelola data permintaan barang yaitu menyunting, menghapus dan memproses data permintaan barang yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.5.

Tabel 5.5 Tabel Pengujian Fitur Mengelola Permintaan Barang

ID	UJ-UC-004
Referensi Kasus Penggunaan	UC-004
Nama	Pengujian Fitur Mengelola Permintaan Datang
Tujuan Pengujian	Menguji kemampuan aplikasi mengelola permintaan datang
Skenario 1	***Aktor menyunting data Good Issue***
Kondisi Awal	Aktor berada pada halaman utama Good Issue IWM
Data Uji	Perubahan data pada Good Issue :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Good Issue yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 2	***Aktor menghapus data Good Issue***
Kondisi Awal	Aktor berada pada halaman utama Good Issue IWM
Data Uji	-

Langkah Pengujian	Pengguna memilih menghapus salah satu data Good Issue yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Good Issue.
Skenario 3	***Aktor memproses data Good Issue***
Kondisi Awal	Aktor berada pada halaman utama Good Issue IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih memproses salah satu data yang terdapat pada Good Issue dengan menekan tombol Approve. Akan muncul konfirmasi untuk penyelesaian data dimana Aktor akan mengkonfirmasi pemrosesan data.
Hasil Yang Diharapkan	Data yang di proses akan merubah data di sub modul lainnya yang berpengaruh serta status pada data terserbut akan berubah
Hasil Yang Didapat	Data berhasil di proses.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Good Issue.

Pada skenario 3 yang digambarkan pada lampiran Gambar A.6 menampilkan proses melihat halaman utama pada submodul *Good Issue*. Skenario dimulai ketika aktor akan melakukan proses lebih lanjut dengan salah satu data *Good Issue* dimana aktor akan menekan tombol *Approve* yang ada di setiap data dapat dilihat pada Lampiran Gambar A.6. Sistem akan memberikan konfirmasi untuk memproses lebih lanjut. Aktor akan melakukan konfirmasi dan sistem akan memprosesnya lebih lanjut. Skenario akan berakhir ketika tombol tidak dapat ditekan lagi dan status data tersebut berubah menjadi "*Published*".

5.2.6.5 Pengujian Fitur Mengelola Pengiriman Barang di Luar Gudang

Pengujian ini terdiri dari pengujian mengelola data pengiriman barang di luar gudang. Pengujian mengelola data pengiriman barang di luar gudang yaitu menambah, menyunting, menghapus dan memproses data pengiriman barang di luar gudang yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.6.

Tabel 5.6 Tabel Pengujian Fitur Mengelola Pengiriman di Luar Gudang

ID	UJ-UC-005
Referensi Kasus Penggunaan	UC-005
Nama	Pengujian Fitur Mengelola Pengiriman brang di Luar Gudang
Tujuan Pengujian	Menguji kemampuan aplikasi mengelola pengiriman barang di luar gudang
Skenario 1	***Pengguna melihat data Transfer Posting***
Kondisi Awal	Aktor berada pada halaman utama Transfer Posting IWM
Data Uji	Inputan data Transfer Posting :
Langkah Pengujian	Aktor masuk ke halaman membuat data Transfer Posting baru dan memasukkan Data Uji ke dalam basis data.
Hasil Yang Diharapkan	Data yang dimasukkan ke dalam basis data berhasil dimasukkan.
Hasil Yang Didapat	Data yang dimasukkan tersimpan di basis data
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data yang baru saja dimasukkan.
Skenario 2	***Aktor menyunting data Transfer Posting***
Kondisi Awal	Aktor berada pada halaman utama Transfer Posting IWM
Data Uji	Perubahan data pada Transfer Posting :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Transfer Posting yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.

Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 3	***Aktor menghapus data Transfer Posting***
Kondisi Awal	Aktor berada pada halaman utama Transfer Posting IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Transfer Posting yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Transfer Posting.
Skenario 4	***Aktor memproses data Transfer Posting***
Kondisi Awal	Aktor berada pada halaman utama Transfer Posting IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih memproses salah satu data yang terdapat pada Transfer Posting dengan menekan tombol Add to Purchase / Add to Production / Deliver Finish. Akan muncul konfirmasi untuk penyelesaian data dimana Aktor akan mengkonfirmasi pemrosesan data.
Hasil Yang Diharapkan	Data yang di proses akan merubah data di sub modul lainnya yang berpengaruh serta status pada data terserbut akan berubah
Hasil Yang Didapat	Data berhasil di proses.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Transfer Posting.

Pada skenario 4 yang digambarkan pada lampiran Gambar A.7 menampilkan proses melihat halaman utama pada submodul *Transfer*

Postiong. Skenario dimulai ketika aktor akan melakukan proses lebih lanjut dengan salah satu data *Transfer Posting* dimana aktor akan menekan tombol Add to Purchase/ Add to Production/ Deliver Finish yang ada di setiap data. Sistem akan memberikan konfirmasi untuk memproses lebih lanjut. Aktor akan melakukan konfirmasi dan sistem akan memprosesnya lebih lanjut. Skenario akan berakhir ketika tombol tidak dapat ditekan lagi dengan nama tombol menjadi "Transfer Done / Purchase Done/ Production Done/ Item Delivered".

5.2.6.6 Pengujian Fitur Mengelola Perpindahan Barang di Dalam Gudang

Pengujian ini terdiri dari pengujian mengelola data perpindahan barang di dalam gudang. Pengujian mengelola data perpindahan barang di dalam gudang yaitu menyunting, dan menghapus data perpindahan barang di dalam gudang yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.7.

Tabel 5.7 Tabel Pengujian Fitur Perpindahan Barang di Dalam Gudang

ID	UJ-UC-006
Referensi Kasus Penggunaan	UC-006
Nama	Pengujian Fitur Mengelola Perpindahan Barang di Dalam Gudang
Tujuan Pengujian	Menguji kemampuan aplikasi mengelola perpindahan barang di dalam gudang
Skenario 1	***Aktor menyunting data Warehouse Movement***
Kondisi Awal	Aktor berada pada halaman utama Warehouse Movement IWM
Data Uji	Perubahan data pada Warehouse Movement :
Langkah Pengujian	Pengguna memilih menyunting salah satu data Warehouse Movement yang ada pada sistem, lalu menyunting data tersebut pada halaman menyunting dan menyimpannya kembali ke basis data.
Hasil Yang Diharapkan	Data yang disunting akan berubah dan disimpan kembali ke basis data.
Hasil Yang Didapat	Data yang disunting tersimpan di basis data
Hasil Pengujian	Berhasil.

Kondisi Akhir	Pengguna berada pada halaman melihat data yang baru saja disunting.
Skenario 2	***Aktor menghapus data Warehouse Movement***
Kondisi Awal	Aktor berada pada halaman utama Warehouse Movement IWM
Data Uji	-
Langkah Pengujian	Pengguna memilih menghapus salah satu data Warehouse Movement yang ada pada sistem, lalu melakukan konfirmasi penghapusan data tersebut dan data akan dihapus oleh sistem.
Hasil Yang Diharapkan	Data yang dihapus akan terhapus di basis data
Hasil Yang Didapat	Data berhasil di hapus.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Pengguna berada pada halaman utama Warehouse Movement.

5.2.6.7 Pengujian Fitur Melihat Stok di Gudang

Pengujian ini terdiri dari pengujian melihat data stok di gudang. Pengujian melihat stok di gudang yaitu melihat data data barang di gudang yang dalam status "Unrestricted Use" yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.8.

Tabel 5.8 Tabel Pengujian Fitur Melihat Stok di Gudang

ID	UJ-UC-007
Referensi Kasus Penggunaan	UC-007
Nama	Pengujian Fitur Melihat Stok di Gudang
Tujuan Pengujian	Menguji kemampuan aplikasi melihat stok di gudang
Skenario 1	***Pengguna melihat data stok di gudang***
Kondisi Awal	Aktor berada pada halaman utama Stok IWM
Data Uji	Inputan data Stok
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data

Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.6.8 Pengujian Fitur Melihat Item Master

Pengujian ini terdiri dari pengujian melihat data *Item Master*. Pengujian melihat *Item Master* yaitu melihat data *Item Master* yang terdiri dari submodul *Item Type, Item Master, Item Detail* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.9.

Tabel 5.9 Tabel Pengujian Melihat Item Master

ID	UJ-UC-008
Referensi Kasus Penggunaan	UC-008
Nama	Pengujian Fitur Melihat Item Master
Tujuan Pengujian	Menguji kemampuan aplikasi melihat item master
Skenario 1	***Pengguna melihat data Item Type***
Kondisi Awal	Aktor berada pada halaman utama Item Type IWM
Data Uji	Inputan data Item Type
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan
Skenario 2	***Pengguna melihat data Item Master***
Kondisi Awal	Aktor berada pada halaman utama Item Master IWM
Data Uji	Inputan data Item Master

Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan
Skenario 3	***Pengguna melihat data Item Detail***
Kondisi Awal	Aktor berada pada halaman utama Item Detail IWM
Data Uji	Inputan data Detail
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.6.9 Pengujian Fitur Melihat Warehouse Master

Pengujian ini terdiri dari pengujian melihat data *Warehouse Master*. Pengujian melihat *Warehouse Master* yaitu melihat data *Warehouse Master* yang terdiri dari submodul *Warehouse, Inventory, Storage, Storage Detail, Bin, Quant* yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.10.

Tabel 5.10 Tabel Pengujian Fitur Melihat Warehouse Master

ID	UJ-UC-009
Referensi Kasus Penggunaan	UC-009
Nama	Pengujian Fitur Melihat Warehouse Master
Tujuan Pengujian	Menguji kemampuan aplikasi melihat warehouse master
Skenario 1	***Pengguna melihat data Warehouse***

Kondisi Awal	Aktor berada pada halaman utama Warehouse IWM
Data Uji	Inputan data Warehouse
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan
Skenario 2	***Pengguna melihat data Inventory***
Kondisi Awal	Aktor berada pada halaman utama Inventory IWM
Data Uji	Inputan data Inventory
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan
Skenario 3	***Pengguna melihat data Storage***
Kondisi Awal	Aktor berada pada halaman utama Storage IWM
Data Uji	Inputan data Storage
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan
Skenario 4	***Pengguna melihat data Storage Detail***
Kondisi Awal	Aktor berada pada halaman utama Storage Detail IWM

Data Uji	Inputan data Storage Detail
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan
Skenario 5	***Pengguna melihat data Bin***
Kondisi Awal	Aktor berada pada halaman utama Bin IWM
Data Uji	Inputan data Bin
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan
Skenario 6	***Pengguna melihat data Quant***
Kondisi Awal	Aktor berada pada halaman utama Quant IWM
Data Uji	Inputan data Quant
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.6.10 Pengujian Fitur Melihat Barang Datang

Pengujian ini terdiri dari pengujian melihat data barang datang. Pengujian melihat barang datang yaitu melihat data barang datang baik

hasil pembelian maupun produksi yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.11.

Tabel 5.11 Tabel Pengujian Melihat Barang Datang

ID	UJ-UC-010
Referensi Kasus Penggunaan	UC-010
Nama	Pengujian Fitur Melihat Barang Datang
Tujuan Pengujian	Menguji kemampuan aplikasi melihat barang datang
Skenario 1	***Pengguna melihat data barang datang***
Kondisi Awal	Aktor berada pada halaman utama Good Receipt IWM
Data Uji	Inputan data Good Receipt
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.6.11 Pengujian Fitur Melihat Permintaan Barang

Pengujian ini terdiri dari pengujian melihat data permintaan barang. Pengujian melihat permintaan barang yaitu melihat data permintaan barang baik dari penjualan, produksi, maupun aset yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.12.

Tabel 5.12 Tabel Pengujian Melihat Permintaan Barang

ID	UJ-UC-011
Referensi Kasus Penggunaan	UC-011
Nama	Pengujian Fitur Melihat Permintaan Barang
Tujuan Pengujian	Menguji kemampuan aplikasi melihat permintaan barang

Skenario 1	***Pengguna melihat data permintaan barang***
Kondisi Awal	Aktor berada pada halaman utama Good Issue IWM
Data Uji	Inputan data Good Issue
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.6.12 Pengujian Fitur Melihat Pengiriman Barang di Luar Gudang

Pengujian ini terdiri dari pengujian melihat data pengiriman barang di luar gudang. Pengujian melihat pengiriman barang di luar gudang yaitu melihat data pengiriman barang ke luar gudang baik pengiriman kepada kustomer atau produksi yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.13.

Tabel 5.13 Tabel Pengujian Fitur Melihat Pengiriman Barang di Luar Gudang

ID	UJ-UC-012
Referensi Kasus Penggunaan	UC-012
Nama	Pengujian Fitur Melihat Pengiriman Barang di Luar Gudang
Tujuan Pengujian	Menguji kemampuan aplikasi melihat pengiriman barang di luar gudang
Skenario 1	***Pengguna melihat data pengiriman barang di luar gudang***
Kondisi Awal	Aktor berada pada halaman utama Good Transfer IWM
Data Uji	Inputan data Good Transfer
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan

Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.6.13 Pengujian Fitur Melihat Perpindahan Barang di Dalam Gudang

Pengujian ini terdiri dari pengujian melihat perpindahan barang di dalam gudang. Pengujian melihat perpindahan barang di dalam gudang yaitu melihat data perpindahan di dalam gudang yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.14.

Tabel 5.14 Tabel Pengujian Fitur Melihat Perpindahan Barang di Dalam Gudang

ID	UJ-UC-013
Referensi Kasus Penggunaan	UC-013
Nama	Pengujian Fitur Melihat Perpindahan Barang di Dalam Gudang
Tujuan Pengujian	Menguji kemampuan aplikasi melihat perpindahan barang di dalam gudang
Skenario 1	***Pengguna melihat data perpindahan barang di dalam gudang***
Kondisi Awal	Aktor berada pada halaman utama Warehouse Movement IWM
Data Uji	Inputan data Warehouse Movement
Langkah Pengujian	Aktor masuk ke halaman melihat data dari salah satu data yang tersimpan di basis data
Hasil Yang Diharapkan	Tertampil detail data dari data pilihan
Hasil Yang Didapat	Data yang dipilih tertampil datanya secara detail
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman melihat data pilihan

5.2.6.14 Pengujian Fitur Melihat Report Jumlah Perpindahan Barang

Pengujian ini terdiri dari pengujian melihat report jumlah perpindahan barang. Pengujian melihat jumlah perpindahan barang yaitu melihat grafik jumlah perpindahan barang tertentu yang sudah ada pada sistem. Rincian skenario pengujian pada kasus penggunaan ini dapat dilihat pada Tabel 5.15.

Tabel 5.15 Tabel Pengujian Fitur Melihat Report Jumlah Perpindahan Barang

ID	UJ-UC-014
Referensi Kasus Penggunaan	UC-014
Nama	Pengujian Fitur Melihat Report Jumlah Perpindahan Barang
Tujuan Pengujian	Menguji kemampuan aplikasi melihat report jumlah perpindahan barang
Skenario 1	***Pengguna melihat report jumlah perpindahan barang***
Kondisi Awal	Aktor berada pada halaman utama IWM
Data Uji	Data Warehouse Movement IWM
Langkah Pengujian	Aktor memilih salah satu item master yang terdapat pada sistem
Hasil Yang Diharapkan	Menampilkan report dari item pilihan
Hasil Yang Didapat	Tertampil report dari item pilihan
Hasil Pengujian	Berhasil
Kondisi Akhir	Aktor berada pada halaman utama

5.3 Evaluasi Pengujian

Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat pada Tabel 6.12. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik disimpulkan bahwa fungsionalitas dari program telah bisa bekerja sesuai dengan yang diharapkan.

Tabel 5.16 Rangkuman Hasil Pengujian

ID	Nama	Skenario	Hasil
UJ-UC-001	Pengujian fitur mengelola *Item Master*	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
		Skenario 4	Berhasil
		Skenario 5	Berhasil
		Skenario 6	Berhasil
		Skenario 7	Berhasil
		Skenario 8	Berhasil
		Skenario 9	Berhasil
		Skenario 10	Berhasil
		Skenario 11	Berhasil
		Skenario 12	Berhasil
UJ-UC-002	Pengujian fitur mengelola *Warehouse Master*	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
		Skenario 4	Berhasil
		Skenario 5	Berhasil
		Skenario 6	Berhasil
		Skenario 7	Berhasil
		Skenario 8	Berhasil
		Skenario 9	Berhasil
		Skenario 10	Berhasil
		Skenario 11	Berhasil
		Skenario 12	Berhasil
		Skenario 13	Berhasil
		Skenario 14	Berhasil
		Skenario 15	Berhasil

ID	Nama	Skenario	Hasil
		Skenario 16	Berhasil
		Skenario 17	Berhasil
		Skenario 18	Berhasil
UJ-UC-003	Pengujian Fitur Mengelola Barang Datang	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
		Skenario 4	Berhasil
UJ-UC-004	Pengujian Fitur Mengelola Permintaan Barang	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
		Skenario 4	Berhasil
UJ-UC-005	Pengujian Fitur Mengelola Pengiriman Barang di Luar Gudang	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
		Skenario 4	Berhasil
UJ-UC-006	Pengujian Fitur Mengelola Perpindahan Barang di Dalam Gudang	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
UJ-UC-007	Pengujian Fitur Melihat Stok di Gudang	Skenario 1	Berhasil
UJ-UC-008	Pengujian Fitur Melihat Data *Item Master*	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
		Skenario 4	Berhasil
UJ-UC-009	Pengujian Fitur Melihat Data *Warehouse Master*	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
		Skenario 4	Berhasil

ID	Nama	Skenario	Hasil
		Skenario 5	Berhasil
		Skenario 6	Berhasil
UJ-UC-010	Pengujian Fitur Melihat Data Barang Datang	Skenario 1	Berhasil
UJ-UC-011	Pengujian Fitur Melihat Data Permintaan Barang	Skenario 1	Berhasil
UJ-UC-012	Pengujian Fitur Melihat Data Pengiriman Barang di Luar Gudang	Skenario 1	Berhasil
UJ-UC-013	Pengujian Fitur Melihat Data Perpindahan Barang di Dalam Gudang	Skenario 1	Berhasil
UJ-UC-014	Pengujian Fitur Melihat Report Jumlah Perpindahan Barang	Skenario 1	Berhasil
UJ.UC.RBAC	Pengujian mengelola *Role Based Access Control (RBAC).*	Skenario 1	Berhasil
		Skenario 2	Berhasil
		Skenario 3	Berhasil
UJ.MTC	Mengelola *Multitenancy*	Skenario 1	Berhasil

LAMPIRAN

Kode Sumber

Dalam memproses *Receipt*, dilakukan pengecekan catatan barang rusak pada *receipt* tersebut untuk menentukan proses yang akan dilakukan berikutnya. Fungsi actionCheckDefect() bertugas untuk mencari catatan barang rusak pada *receipt* yang akan diproses. Jika pada *receipt* tidak ditemukan barang rusak (*defect*) maka akan dilakukan proses normal yang dilakukan oleh fungsi actionFromPurchase (jika *receipt* dari pembelian) atau actionFromProduction (jika *receipt* dari hasil produksi). Proses normal hanya akan mengubah status dari *receipt* tersebut menjadi 3 dimana menandakan *receipt* tersebut sudah selesai dijalankan. Jika terdapat barang rusak pada *receipt* maka akan diberikan beberapa pilihan yang dapat dilihat pada Lampiran gambar A.6. Tombol Dump Item akan menjalankan fungsi actionDumpPurchase() atau actionDumpProduction() dimana fungsi ini akan menjalankan proses membuang barang yang rusak dan mengubah status *receipt* menjadi 3. Tombol Ignore Item akan menjalankan fungsi actionUpdatePurchase() atau actionUpdateProduction() dimana membiarkan barang rusak masuk ke dalam gudang dan mengganti status *receipt* menjadi 3. Tombol Return Item (Jika *receipt* dari pembelian)/ Reproduct Item (Jika *receipt* dari hasil produksi) akan menjalankan fungsi actionNewPurchase() atau actionNewProduction() dimana fungsi ini akan membuat order produksi kepada divisi produksi atau permintaan barang lagi kepada divisi pembelian untuk mengganti barang yang rusak dan menghapus barang yang rusak tersebut serta mengganti status *receipt* menjadi 3.

```
public function actionCheckDefect($id){
    $receipt = IwmGoodReceiptH::findOne($id);
    $model = IwmGoodReceiptF::find()->where('gr_id = '.$id.' and
defect > 0')->all();
```

```
        return count($model) == 0 ? 1 : 0;}

    public function actionPurchaseDefect($id){
        IwmgoodreceipthController::actionDeleteDefect($id);
        $another = IwmGoodReceiptH::find()->where('from_po_id =
'.$receipt['from_po_id'].' and id != '.$receipt['id'])->all();
        foreach ($another as $key => $value) {
            $value->update();}
        $pur = PurPoHeader::findOne($receipt['from_po_id']);
        $pur->update();
        return $this->redirect(['index']);    }

    public function actionDefect($id)
    {
        $receipt = IwmGoodReceiptH::findOne($id);
        $model = IwmGoodReceiptF::find()->where('gr_id = '.$id.' and
defect > 0')->all();
        $providerIwmGoodReceiptF = new
\yii\data\ArrayDataProvider([
            'allModels' => $model,
        ]);
        if($name[0] == "Production" || $name[0] ==
"ReturnProduction"){
            return $this->renderAjax('_production', [
                'model' => $providerIwmGoodReceiptF,
                'id' => $id,
            ]);}
        else if ($name[0] == "Purchase"){
            return $this->renderAjax('_purchase', [
                'model' => $providerIwmGoodReceiptF,
                'id' => $id,
            ]);}
        else if ($name[0] == "Return"){
            $receipt->update();
            $issue = IwmGoodIssueH::find()->where('to_so_id =
'.$name[1])->one();
```

```
        foreach ($model as $key => $value) {
            $issue_line = IwmGoodIssueF::find()
                ->where('gi_id = '.$issue['id'].'
                    and item_id = '.$value['item_id'])->one();
            $quant = IwmQuant::find()
                ->where('item_id in (
                    select id
                    from iwm_item_detail
                    where item_id = '.$value['item_id'].'
                    )
                and status = "Unrestricted Use"')->all();
            foreach ($quant as $key2 => $value2) {
                if ($temp > 0) {
                    if($temp >= $value2['item_qty']){}
                    else if ($temp < $value2['item_qty']){}
IwmwarehousemovementController::actionCreateMovement($value
['id'], null, $wm_name, $date_taken, null, $from_storage_id,
$from_sd_id, $from_bin_id, $sd_id, null, $to_storage_id, $to_sd_id,
$to_bin_id, $selected_quant['item_id'], $uom['uom_id'],
$selected_quant['item_qty']);}
                else if ($temp == 0){
                    $issue->update();}
                $new_qty =
IwmquantController::actionGetFreeQtyDetailItem($value2['item_id'
], $inv['id']);
                $item_data =
IwmItemDetail::findOne($value2['item_id']);
                $item_data['qty_in_inventory'] = $new_qty;
                $item_data->update();

                $stock = IwmStock::find()
                    ->where('item_id = (
                        select item_id
                        from iwm_item_detail
                        where id = '.$value2['item_id'].'
                    )')->one();
```

```
                $stock['current_qty'] =
IwmquantController::actionReturnQty($stock['item_id']);
                $stock->update();
            }
            $issue_line->update();
            if($temp > 0){
                $pro_request = new PpProductionRequest();
                $pro_request->save();

                $get_conn = new IwmSdProConn();
                $get_conn->save();

                $production_material = new
ProductionMaterial($pro_request->material_produced,
$pro_request->id);
                $production_material->automateProductionMaterial();
                $production_material->insertProductionMaterial();
            }
         }
        $check_line = IwmGoodIssueF::find()->where('gi_id =
'.$issue['id'].'
            and item_demand > qty_in_inventory')->all();
        if(count($check_line) == 0){
            $issue->update();
        }
        return $this->redirect(['index']);
      }
   }

   public function actionNewPurchase($id)
   {
      IwmgoodreceipthController::actionDeleteDefect($id);
      IwmgoodreceipthController::actionFromPurchase($id);
      IwmgoodreceipthController::actionCreatePurchase($id);
      return $this->redirect(['index']);
   }
```

```
    public function actionDumpPurchase($id){
        IwmgoodreceipthController::actionDeleteDefect($id);
        IwmgoodreceipthController::actionFromPurchase($id);
        return $this->redirect(['index']);
    }
    public function actionUpdatePurchase($id){
        IwmgoodreceipthController::actionFromPurchase($id);
        return $this->redirect(['index']);
    }

    public function actionNewProduction($id)
    {
        IwmgoodreceipthController::actionCreateProduction($id);
        IwmgoodreceipthController::actionDeleteDefect($id);
        IwmgoodreceipthController::actionFromProduction($id);
        return $this->redirect(['index']);
    }
    public function actionDumpProduction($id){
        IwmgoodreceipthController::actionDeleteDefect($id);
        IwmgoodreceipthController::actionFromProduction($id);
        return $this->redirect(['index']);
    }
    public function actionUpdateProduction($id){
        IwmgoodreceipthController::actionFromProduction($id);
        return $this->redirect(['index']);
    }

    public function actionCreatePurchase($id)
    {
        $model = $this->findModel($id);
        $poh = PurPoHeader::findOne($model['from_po_id']);
        $model->update();
        $poh->update();
    }

    public function actionCreateProduction($id)
```

```
    {
        $model = $this->findModel($id);
        $line = IwmGoodReceiptF::find()->where('gr_id = '.$id)-
>one();
        $pref_pro                                                    =
PpProductionOrder::findOne($model['from_production_id']);
        $pref_rec = PpProductionRequest::find()->where('id =
'.$pref_pro['production_request_id'])->one();
        $pro_request = new PpProductionRequest();
        $pro_request->save();
        $production_material = new ProductionMaterial($pro_request-
>material_produced, $pro_request->id);
        $production_material->automateProductionMaterial();
        $production_material->insertProductionMaterial();
    }
    public function actionFromPurchase($id)
    {
        $model = $this->findModel($id);
        $poh = PurPoHeader::find()
            ->where('id = '.$model['from_po_id'])->One();
        $poh->update();

        $update_receipt = IwmGoodReceiptH::findOne($model['id']);
        $update_receipt->update();

        if($update_receipt['to_production_id'] != null){
            $issue = IwmGoodIssueH::find()
                ->where('to_production_id in (
                    select production_order_id
                    from pp_production_material
                    where production_request_id in (
                        select production_request_id
                        from pp_production_order
                        where id = '.$update_receipt['to_production_id'].')
                    )
                and issue_status = 0')->all();
```

```
        if($issue != null){
            foreach ($issue as $key => $value) {
                $line = IwmGoodIssueF::find()
                    ->where('gi_id = '.$value['id'].'
                        and item_demand > qty_in_inventory')->all();
                if(count($line) == 0){
                    $value->update();
                }}}}}
    public function actionFromProduction($id)
    {
        $model = $this->findModel($id);
        $model->update();

        $update_receipt = IwmGoodReceiptH::findOne($model['id']);
        $update_receipt->update();

        if ($update_receipt['to_production_id'] != null) {
            $issue = IwmGoodIssueH::find()
            ->where('to_production_id in (
                select production_order_id
                from pp_production_material
                where production_request_id in (
                    select production_request_id
                    from pp_production_order
                    where id = '.$update_receipt['to_production_id'].')
                )
            and issue_status = 0')->all();
        if($issue != null){
            foreach ($issue as $key => $value) {
                $line = IwmGoodIssueF::find()
                    ->select('F.*')
                    ->from('iwm_good_issue_h as H, iwm_good_issue_f
as F')
                    ->where('F.gi_id = H.id
                        and F.gi_id in (
                            select id
```

```
                        from iwm_good_issue_h
                        where to_production_id in (
                            select distinct production_order_id
                            from pp_production_material
                            where production_request_id = (
                                select production_request_id
                                from pp_production_order
                                where id =
'.$update_receipt['to_production_id'].'
                                )
                            )
                        and issue_status = 0
                        )
                    and F.item_demand > F.qty_in_inventory')->all();
                if(count($line) == 0){
                    $value->update();
}}}}}
    public function actionDeleteDefect($id)
    {
        $wm = IwmWarehouseMovement::find()
            ->select('W.*')
            ->from('iwm_warehouse_movement as W, iwm_storage as
S')
            ->where('W.to_storage_id = S.id
                and S.storage_type = "Defect"
                and W.gr_id in (
                    select id
                    from iwm_good_receipt_f
                    where gr_id = '.$id.'
                    )')->all();
        foreach ($wm as $key => $value) {
            $quant = IwmQuant::find()
                ->where('bin_id = '.$value['from_bin_id'].'
                    and item_id = '.$value['item_id'].'
                    and status = "GR Blocked Stock"')->all();
            foreach ($quant as $key2 => $value2) {
```

```
            $value2->delete();
        }
        $value->delete();
    }}
  public function actionCreateReceipt($type, $id, $date, $item = array()){
    $receipt = new IwmGoodReceiptH();
    if($type == "Purchase"){
        $inv = IwmInventory::find()->select('I.id')
            ->from('iwm_inventory as I, iwm_inventory_type as T, iwm_item_type as IT')
            ->where('T.inventory_id = I.id
                and T.item_type = IT.id
                and IT.item_type_name = "Raw"')->one();
        $find_pur = IwmGoodReceiptH::find()->where('from_po_id = '.$id)->one();
        if($find_pur != null){
            $pur = PurPoHeader::findOne($id);
            $pur->update();
        }
        else{
        }
        $receipt->save();
        foreach ($item as $key => $value) {
            $line = new IwmGoodReceiptF();
            $line->save();

            $po = PurPoLine::find()
                ->where('po_header_id = '.$pur['id'].'
                and item_number = '.$value['item_id'])->one();
            $checking_item = IwmItemDetail::find()
                ->where('po_id = '.$po['po_header_id'].'
                    and po_line_id = '.$po['id'].'
                    and item_id = '.$value['item_id'])->one();
            if ($checking_item != null) {
                $checking_item->update();
```

```
            }
            else if ($checking_item == null){
               $checking_item = new IwmItemDetail();
               $checking_item->save(); }}}
      else if ($type == "Production"){
         $find_another = PpProductionMaterial::find()
            ->select('M.*')
            ->from('pp_production_order as O,
pp_production_material as M')
            ->where('M.production_order_id = O.id
               and O.production_request_id = (
                  select production_request_id
                  from pp_production_order
                  where id = '.$id.')
               and M.material_id = '.$data['id'])->one();
         if($find_another != null){
            $receipt['to_production_id'] =
$find_another['production_order_id'];
         }
         $receipt->save();
         $item_send = array();
         foreach ($item as $key => $value) {
            $line = new IwmGoodReceiptF();
            $line->save();

            $data = array(
               'item_id' => $value['item_id'],
               'item_qty' => $value['item_qty']
            );
            array_push($item_send, $data);
         }

IwmtransferpostinghController::actionCreateTransfer("Production",
$id, $date, $item_send);
      }
      else if ($type == "Return"){
```

```
            $receipt->save();

            $issue = new IwmGoodIssueH();
            $issue->save();

            foreach ($item as $key => $value) {
                $line = new IwmGoodReceiptF();
                $line->save();

                $issue_line = new IwmGoodIssueF();
                $issue_line->save();
            }
        }
        return $receipt;
    }
```

Kode Sumber A.1 IwmGoodReceiptHController

Dalam memproses *Issue*, harus diawali dengan menyelesaikan *receipt* terlebih dahulu untuk pelaku yang sama. Fungsi actionUpdateIssue() akan melakukan perubahan status terhadap barang yang dipesan menjadi "*In Quality Inspection*" dan menyiapkan barang untuk proses pengiriman.

```
    public function actionUpdateIssue($id)
    {
        $model = $this->findModel($id);
        $model['issue_date'] = date("Y-m-d H:i:s");
        $model->update();
        $inv = $model['to_inventory_id'];
        $issue_name = explode("-",$model['issue_name']);
        $item_send = array();
        $query = IwmGoodIssueF::find()
            ->select('F.*')
            ->from('iwm_good_issue_f as F, iwm_good_issue_h as H')
            ->where('F.gi_id = H.id and H.id = '.$model['id'])->all();
        foreach ($query as $key) {
            $data = array(
```

```
            'item_id' => $key['item_id'],
            'item_qty' => $key['item_demand'],
        );
        array_push($item_send, $data);
        $reserve = $key['item_demand'];
        $item = $key['item_id'];

        $this->actionReserve($model['id'], $model['issue_date'],
$key['id'], $key['from_inventory_id'], $model['to_so_id'],
$model['to_production_id'], $inv, $item, $reserve);
        if ($model['to_so_id'] != null){
            SdSalesitemController::actionPick($model['to_so_id'],
$model['issue_date'], $item_send);
        }
    }
    if($issue_name[1] == "Asset"){
        $asset = AmAssetReqH::findOne($issue_name[2]);
        $asset['status'] = "Send";
        $asset->update();

        $quant = IwmQuant::find()
            ->where('status = "Reserved"
                and sales_id is null
                and production_id is null')->all();
        foreach ($quant as $key => $value) {
            $value['status'] = "In Quality Inspection";
            $value->update();


IwmbinController::actionChangeStatus($value['bin_id'],
$value['item_id']);
        }
    }
    else if ($model['to_production_id'] != null){
```

```
        $goodsReceive = PpProductionGoodsReceive::find()-
>where("production_order_id = ".$model['to_production_id'])-
>one();
        $productionMaterial = PpProductionMaterial::find()-
>where("production_order_id = ".$model['to_production_id'])-
>all();
        foreach ($productionMaterial as $material) {
           $material->goods_receive_id = $goodsReceive->id;
           $material->save();
        }

        $quant = IwmQuant::find()
           ->where('status = "Reserved"
              and sales_id is null
              and production_id = '.$model['to_production_id'])-
>all();
        foreach ($quant as $key => $value) {
            $value['status'] = "In Quality Inspection";
            $value->update();

IwmbinController::actionChangeStatus($value['bin_id'],
$value['item_id']);
        }
      }
      else{
        if ($model['to_so_id'] != null){

SdSalesitemController::actionConfirm($model['to_so_id'],
$model['issue_date'], $item_send);
            $quant = IwmQuant::find()
               ->where('status = "Reserved"
                  and sales_id = '.$model['to_so_id'].'
                  and production_id is null')->all();
            foreach ($quant as $key => $value) {
                $value['status'] = "In Quality Inspection";
                $value->update();
```

```
IwmbinController::actionChangeStatus($value['bin_id'],
$value['item_id']);
            }
          }
          if ($model['issue_type'] == 1 && $model['to_so_id'] ==
null){

IwmtransferpostinghController::actionCreateTransfer("Inventory",
$model['id'], $model['issue_date'], $item_send);
          }
       }
       $update_issue = IwmGoodIssueH::findOne($model['id']);
       $update_issue['issue_status'] = 3;
       $update_issue->update();
       return $this->redirect(['index']);
    }
```

Kode Sumber A.2 IwmGoodIssueHController

Dalam memproses *Transfer*, diasumsikan satu catatan *transfer* ini adalah satu kali pengiriman yang datang. Fungsi actionUpdateIssue(), actionUpdatePurchase() dan actionUpdateProduction(). Adalah 3 fungsi yang hampir sama yakni actionUpdateIssue() memproses barang pada saat truk kembali ke gudang, proses ini menyatakan bahwa barang sudah dikirim ke pemesan. Fungsi ini menghapus barang dengan status "*Delivering*" dan lalu memberikan informasi kepada bagian penjualan bahwa truk sudah siap digunakan untuk pengiriman berikutnya. actionUpdatePurchase() memproses barang datang dari hasil pembelian, fungsi ini membuat proses penataan barang di gudang secara FIFO dengan membuat *Item Detail* dan *Quant* serta *Log Movement* barang dari truk ke *Bin* yang sudah disiapkan. actionUpdateProduction() memproses barang yang datang dari hasil produksi. Fungsi ini melakukan pekerjaan yang sama seperti actionUpdatePurchase() hanya saja berbeda sumber.

```
public function actionUpdateIssue($id)
```

```
    {
        $model = $this->findModel($id);
        $model['date_shipped'] = date("Y-m-d H:i:s");
        $delivery = explode("-",$model['tp_name']);
        $delivery_id = $delivery[2];
        SdDeliveryController::actionCompleteTransport($delivery_id);
        $model->update();
        $issue = IwmGoodIssueH::findOne($model['gi_id']);
        $issue_line = IwmGoodIssueF::find()
            ->where('gi_id = '.$model['gi_id'])->all();
        $count = 0;
        foreach ($issue_line as $key => $value) {
            $transfer_item = IwmTransferPostingF::find()
                ->select('F.item_qty')
                ->from('iwm_transfer_posting_h as H,
iwm_transfer_posting_f as F')
                ->where('F.tp_id = H.id
                    and H.gi_id in (
                        select H.id
                        from iwm_good_issue_h as H, iwm_good_issue_f as
F
                        where F.gi_id = h.id
                        and F.item_id = '.$value['item_id'].'
                        and H.id = '.$model['gi_id'].')')->sum('F.item_qty');
            if($transfer_item == $value['item_demand']){
                $count++;
            }
        }
        if (count($issue_line) == $count){
            $issue['issue_status'] = 4;
            $issue->update();
            foreach ($issue_line as $key => $value) {

IwmQuantController::actionDeleteQuant($issue['to_so_id'],null,$va
lue['item_id']);
            }
```

```
        }

        $transfer = IwmTransferPostingH::findOne($model['id']);
        $transfer['transfer_status'] = 1;
        $transfer->update();
        return $this->redirect(['index']);
    }
  public function actionUpdatePurchase($id)
    {
        $model = $this->findModel($id);
        $po = PurPoHeader::findOne($model['po_id']);
        $ro = PurRoHeader::find()->where('po_header_id =
'.$model['po_id'])->one();
        if($ro != null){
            $ro_line = PurRoLine::find()
                ->where('ro_header_id = '.$ro['id'])->all();
            $get_receipt = IwmGoodReceiptH::find()
                ->where('from_po_id = '.$model['po_id'])->all();
            $receipt = null;
            foreach ($get_receipt as $key => $value) {
                $name = explode("-", $value['receipt_name']);
                if($name[0] == "ReturnPurchase"){
                    break;
                }
            }
            $count = 0;
            foreach ($ro_line as $key => $value) {
                $data_transfer = IwmTransferPostingF::find()
                    ->select('F.*')
                    ->from('iwm_transfer_posting_h as H,
iwm_transfer_posting_f as F')
                    ->where('F.tp_id = H.id
                        and H.po_id = '.$receipt['from_po_id'].'
                        and F.item_id = '.$value['item_number'].'
                        and H.transfer_status = 1')->all();
                foreach ($data_transfer as $key2 => $value2) {
```

```
                $header = IwmTransferPostingH::findOne($value2['tp_id']);
                if($name[0] == "TransferReturn"){
                }
            }
            $transfer_item = IwmTransferPostingF::find()
                ->select('F.item_qty')
                ->from('iwm_transfer_posting_h as H, iwm_transfer_posting_f as F')
                ->where('F.tp_id = H.id
                    and H.po_id = '.$receipt['from_po_id'].'
                    and H.id = '.$model['id'].'
                    and F.item_id = '.$value['item_number'])->one();
            if($transfered_item+$transfer_item['item_qty'] == $value['quantity']){
            }
            else if ($transfered_item+$transfer_item['item_qty'] > $value['quantity']){
                $receipt_line = IwmGoodReceiptF::find()->where('gr_id = '.$receipt['id'].'
                    and item_id = '.$value['item_number'])->one();
                $receipt_line->update();
            }
        }
        if (count($ro_line) == $count){
            $receipt->update();
            $po->update();
        }
    }
    else if ($ro == null){
        $po_line = PurPoLine::find()
            ->where('po_header_id = '.$model['po_id'])->all();
        $receipt = IwmGoodReceiptH::find()
            ->where('from_po_id = '.$model['po_id'])->one();
        $count = 0;
        foreach ($po_line as $key => $value) {
```

```
            $transfered_item = IwmTransferPostingF::find()
               ->select('F.item_qty')
               ->from('iwm_transfer_posting_h as H,
iwm_transfer_posting_f as F')
               ->where('F.tp_id = H.id
                  and H.po_id = '.$value['po_header_id'].'
                  and F.item_id = '.$value['item_number'].'
                  and H.transfer_status = 1')->sum('F.item_qty');
            $transfer_item = IwmTransferPostingF::find()
               ->select('F.item_qty')
               ->from('iwm_transfer_posting_h as H,
iwm_transfer_posting_f as F')
               ->where('F.tp_id = H.id
                  and H.po_id = '.$value['po_header_id'].'
                  and H.id = '.$model['id'].'
                  and F.item_id = '.$value['item_number'])->one();
            if($transfered_item+$transfer_item['item_qty'] ==
$value['quantity']){
               $count++;
            }
            else if ($transfered_item+$transfer_item['item_qty'] >
$value['quantity']){
               $receipt_line = IwmGoodReceiptF::find()-
>where('gr_id = '.$receipt['id'].'
                  and item_id = '.$value['item_number'])->one();
               $receipt_line->update();
            }
         }
         if (count($po_line) == $count){
            $receipt->update();
            $po->update();
         }}
      $transfer_line = IwmTransferPostingF::find()->where('tp_id =
'.$model['id'])->all();
      foreach ($transfer_line as $key => $value) {
```

```
        /* Setting Defect Receipt Line */
        $receipt_line = IwmGoodReceiptF::find()->where('gr_id =
'.$receipt['id'].'
            and item_id = '.$value['item_id'])->one();
        if($receipt_line['defect'] == null){}
        else {}
        $receipt_line->update();

        if($receipt['to_production_id'] != null){
            $issue_line = IwmGoodIssueF::find()
                ->where('item_id = '.$value['item_id'].'
                    and gi_id in (
                        select id
                        from iwm_good_issue_h
                        where to_production_id in (
                            select production_order_id
                            from pp_production_material
                            where production_request_id in (
                                select production_request_id
                                from pp_production_order
                                where id = '.$receipt['to_production_id'].')
                            )
                        )')->one();
            $issue_line->update();
            $issue = IwmGoodIssueH::findOne($issue_line['gi_id']);
        }

        $detail = IwmItemDetail::find()
            ->where('po_id = '.$po['id'].'
                and item_id = '.$value['item_id'])->One();
        if ($ro != null) {}
        $move_defect = new IwmWarehouseMovement();
        if($value['defect'] > 0){
            $def_quant = new IwmQuant();
            $def_quant->save();
```

```
        }
        else if ($value['defect'] <= 0){
        }
        $detail->update();
        while($temp_qty > 0){
           $move = new IwmWarehouseMovement();
           $new_quant = new IwmQuant();
           if($receipt['to_production_id'] != null){
              $production_material = PpProductionMaterial::find()
                 ->select('M.*')
                 ->from('pp_production_order as O,
pp_production_material as M')
                 ->where('M.production_order_id = O.id
                    and O.production_request_id in (
                       select production_request_id
                       from pp_production_order
                       where id = '.$receipt['to_production_id'].')
                    and M.material_id = '.$value['item_id'])->one();
              if($production_material != null){}
           }
           else {
              $sure = IwmInventory::find()->select('I.id')
                 ->from('iwm_inventory as I, iwm_inventory_type as
T, iwm_item_type as IT')
                 ->where('T.inventory_id = I.id
                    and T.item_type = IT.id
                    and IT.item_type_name = "Non Product"')->one();
              if($model['to_inventory_id'] == $sure['id']){
                 $po =
PurPoHeader::findOne($receipt['from_po_id']);
                 $name = explode("-", $po['po_number']);
                 $move_defect['wm_name'] = "WM-Defect-
".$receipt['id']."-".$receipt_line['id']."-Asset-".$name[3]."-
".$po['need_by_date']."-".$detail['item_name'];
```

```
                $move['wm_name'] = "WM-Rec-".$receipt['id']."-
".$receipt_line['id']."-Asset-".$name[3]."-".$po['need_by_date']."-
".$detail['item_name'];
                $new_quant['status'] = "Reserved";

                $issue = null;
                $issue_id = IwmGoodIssueH::find()
                   ->where('id in (
                      select gi_id
                      from iwm_good_issue_f
                      where item_id = (
                         select item_id
                         from iwm_item_detail
                         where id = '.$detail['id'].'
                         ))
                      and issue_status = 0')->all();
                foreach ($issue_id as $key3 => $value3) {
                   if($wm_name[5] == $name_issue[2]){
                      break;
                   }}
                if ($issue != null) {
                   if ($receipt_status == 1) {
                      $issue->update();
                   }}}
             else if ($model['to_inventory_id'] != $sure['id']){}}

          /*
          *    Creating new Movement to move from Receiving
storage to inside storage
          */

          $check_type = IwmInventory::find()
             ->select('I.*')
             ->from('iwm_inventory as I, iwm_inventory_type as T')
             ->where('T.inventory_id = I.id
                and T.item_type = (
```

```
                select id
                from iwm_item_type
                where item_type_name = "Non Product"
                )')->one();
            if($check_type['id'] == $model['to_inventory_id']){}
            else {
                $get_bin = IwmBin::find()
                    ->where('item_id in (
                            select item_id
                            from iwm_item_detail
                            where id = '.$detail['id'].')
                        and bin_status != 2')->all();
                $to_bin = null;
                foreach ($get_bin as $key2 => $value2) {
                    $get_quant = IwmQuant::find()->where('bin_id =
'.$value2['id'])->sum('item_qty');
                    if($get_quant == null){}
                    if(($value2['max_item_placed'] > $get_quant)){
                    }}
                if($to_bin != null){
                    $new_quant['bin_id'] = $to_bin['id'];
                    $quantity = IwmQuant::find()->where('bin_id =
'.$value2['id'])->sum('item_qty');

                    if($quantity == null){}
                    if($quantity > 0){
                        if($to_bin['max_item_placed'] > $quantity){
                            if($temp_qty >= ($to_bin['max_item_placed']-
$quantity)){}
                            else if($temp_qty <
($to_bin['max_item_placed']-$quantity)){}}}
                    else if ($quantity <= 0){
                        if($temp_qty >= $to_bin['max_item_placed']){}
                        else if($temp_qty <
$to_bin['max_item_placed']){}}}
                else if ($to_bin == null){}}
```

```
            $to_bin->update();

            if($value['defect'] > 0){}
            $new_quant->save();
            $move->save();
        }
        if($value['defect'] > 0){
            $def_quant->save();
            $move_defect->save();
        }

        $new_qty = IwmquantController::actionGetFreeQtyDetailItem($value['item_id'], $receipt['to_inventory_id']);
        $item_data = IwmItemDetail::findOne($detail['id']);
        $item_data['qty_in_inventory'] = $new_qty;
        $item_data->update();

        $stock = IwmStock::find()->select('S.*')
            ->from('iwm_stock as S, iwm_item_master as M, iwm_item_detail as ID')
            ->where('S.item_id = M.id
                and M.id = (
                    select item_id
                    from iwm_item_detail
                    where id = '.$detail['id'].'
                    )')->one();
        $stock['current_qty'] = IwmquantController::actionReturnQty($stock['item_id']);
        if($stock['current_qty'] > $stock['max_qty']){}
        $stock->update();
      }
      $transfer = IwmTransferPostingH::findOne($model['id']);
      $transfer->update();
      return $this->redirect(['index']);
    }
```

```
    public function actionUpdateProduction($id)
    {
        $model = $this->findModel($id);
        $receipt = IwmGoodReceiptH::find()
            ->where('from_production_id = '.$name[2])->one();
        $receipt->update();
        $transfer = IwmTransferPostingH::findOne($model['id']);
        $transfer->update();

        $production = PpProductionOrder::findOne($name[2]);
        $item = new IwmItemDetail();
        $name = IwmItemMaster::find()
            ->where('id = '.$production['material_produced'])->One();

        $query = "Select * from fi_cogm where production_order =
'$production[id]'";
        $db = Yii::$app->db;
        $command = $db->createCommand($query);
        $cost = $command->queryOne();
        if($cost != null){
            $item['cost'] = (float)$cost['cogm']/(float)$cost['qty'];
        }
        $item->save();
        $transfer_line = IwmTransferPostingF::find()->where('tp_id =
'.$model['id'])->all();
        foreach ($transfer_line as $key) {
            $receipt_line = IwmGoodReceiptF::find()->where('gr_id =
'.$receipt['id'].'
                and item_id = '.$key['item_id'])->one();
            if($receipt_line['defect'] == null){}
            else {}
            $receipt_line->update();

            if($receipt['to_production_id'] != null){
                $issue_line = IwmGoodIssueF::find()
                    ->where('item_id = '.$key['item_id'].'
```

```
                and gi_id in (
                    select id
                    from iwm_good_issue_h
                    where to_production_id in (
                        select production_order_id
                        from pp_production_material
                        where production_request_id in (
                            select production_request_id
                            from pp_production_order
                            where id = '.$receipt['to_production_id'].')
                        )
                    )')->one();
            $issue_line->update();
            $issue = IwmGoodIssueH::findOne($issue_line['gi_id']);
        }

        $detail = IwmItemDetail::find()
            ->where('production_id = '.$production['id'].'
                and item_id = '.$key['item_id'])->One();
        $move_defect = new IwmWarehouseMovement();
        if($key['defect'] > 0){
            $def_quant = new IwmQuant();
            $def_quant->save();}
        else if ($key['defect'] <= 0){}
        $detail->update();
        $issue_id = null;
        while($temp_qty > 0){
            $move = new IwmWarehouseMovement();
            $new_quant = new IwmQuant();
            /*
            *  Update the quantity value from Item detail after creating
the Receipt
            */
            $detail = IwmItemDetail::find()
                ->where('id = '.$item['id'])->One();
            $detail->update();
```

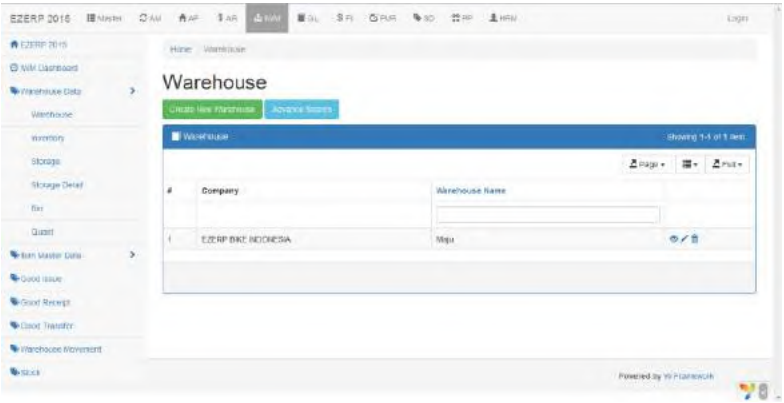
```
            $bins= IwmBin::find()
               ->where('id = '.$bin['id'])->One();
            $bins->update();
            $get_bin = IwmBin::find()
               ->where('item_id in (
                     select M.id
                     from iwm_item_master as M, iwm_item_detail as
ID
                     where ID.item_id = M.id
                     and ID.id = '.$item['id'].')
                  and bin_status != 2')->all();
            $to_bin = null;
            foreach ($get_bin as $key2 => $value2) {
               $get_quant = IwmQuant::find()->where('bin_id =
'.$value2['id'])->sum('item_qty');
               if($get_quant == null){}
               if($value2['max_item_placed'] > $get_quant){}
            }
            if ($to_bin !== null) {}
            if($quantity == null){}
            if($receipt['to_production_id'] != null){
               $production_material = PpProductionMaterial::find()
                  ->select('M.*')
                  ->from('pp_production_order as O,
pp_production_material as M')
                  ->where('M.production_order_id = O.id
                     and O.production_request_id = (
                        select production_request_id
                        from pp_production_order
                        where id = '.$production['id'].')
                     and M.material_id =
'.$production['material_produced'])->one();
               if($production_material != null){}
            }
            else {
```

```
            $get_conn = IwmSdProConn::find()-
>where('production_id = '.$production['production_request_id'])-
>one();
            if($get_conn != null){
               $get_issue_sd = IwmGoodIssueF::find()
                  ->select('F.*')
                  ->from('iwm_good_issue_h as H,
iwm_good_issue_f as F')
                  ->where('F.gi_id = H.id
                     and H.to_so_id = '.$get_conn['sd_id'].'
                     and F.item_id = '.$key['item_id'])->one();
               $get_issue_sd->update();
               $get_issue =
IwmGoodIssueH::findOne($get_issue_sd['gi_id']);}
            else {
               $find_line = IwmGoodIssueF::find()
                  ->where('gi_id = '.$receipt['id'].'
                     and item_id = '.$key['item_id'])->one();
            }}
         if($key['defect'] > 0){}
         /*
         *  Creating new Movement to move from Receiving
storage to inside storage
         */
         if($quantity > 0){
            if($to_bin['max_item_placed'] > $quantity){
               if($temp_qty >= ($to_bin['max_item_placed']-
$quantity)){}
               else if ($temp_qty < ($to_bin['max_item_placed']-
$quantity)){}}}
         else if ($quantity <= 0){
            if($temp_qty >= $to_bin['max_item_placed']){}
            else if($temp_qty < $to_bin['max_item_placed']){}}
         $to_bin->update();
         $new_quant->save();
         $move->save();
```

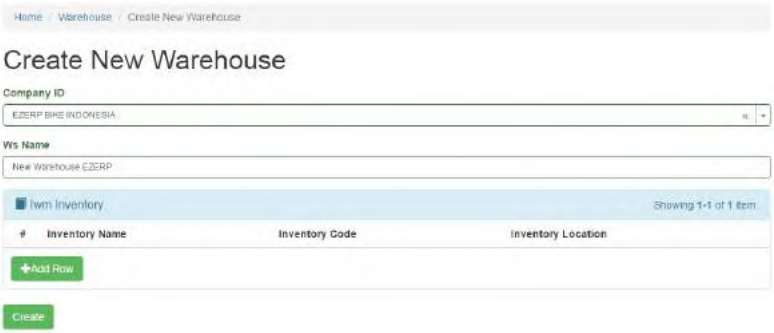
```
        }
        if($issue_id != null){
            $find_line = IwmGoodIssueF::find()
                ->where('gi_id = '.$issue_id.'
                    and item_id = '.$key['item_id'])->one();
            $find_line->update();
        }
        if($key['defect'] > 0){
            $def_quant->save();
            $move_defect->save();
        }
        $new_qty                                                    =
IwmquantController::actionGetFreeQtyDetailItem($key['item_id'],
$receipt['to_inventory_id']);
        $item_data = IwmItemDetail::findOne($item['id']);
        $item_data->update();

        $stock = IwmStock::find()->select('S.*')
            ->from('iwm_stock as S, iwm_item_master as M,
iwm_item_detail as ID')
            ->where('S.item_id = M.id
                and M.id = (
                    select item_id
                    from iwm_item_detail
                    where id = '.$item['id'].'
                    )')->one();
        $stock['current_qty']                                       =
IwmquantController::actionReturnQty($stock['item_id']);
        if($stock['current_qty'] > $stock['max_qty']){
        }
        $stock->update();}
      return $this->redirect(['index']);    }
```

Kode Sumber A.3 IwmTransferPostingHController

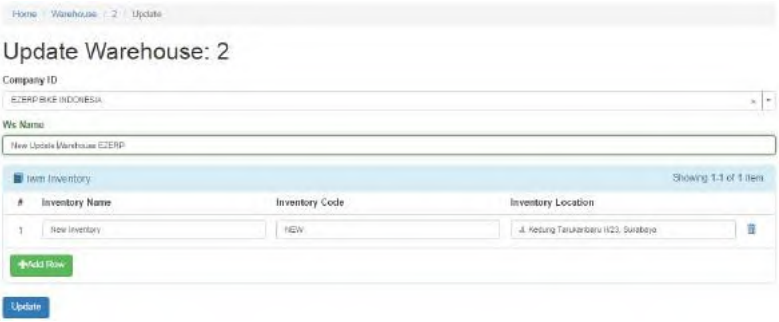
Gambar



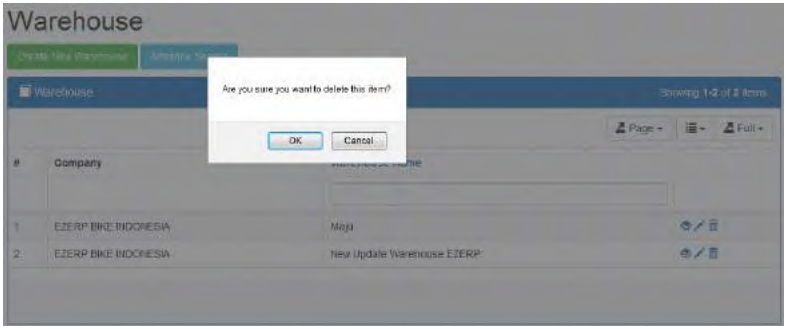
Gambar A.1 View Index Warehouse



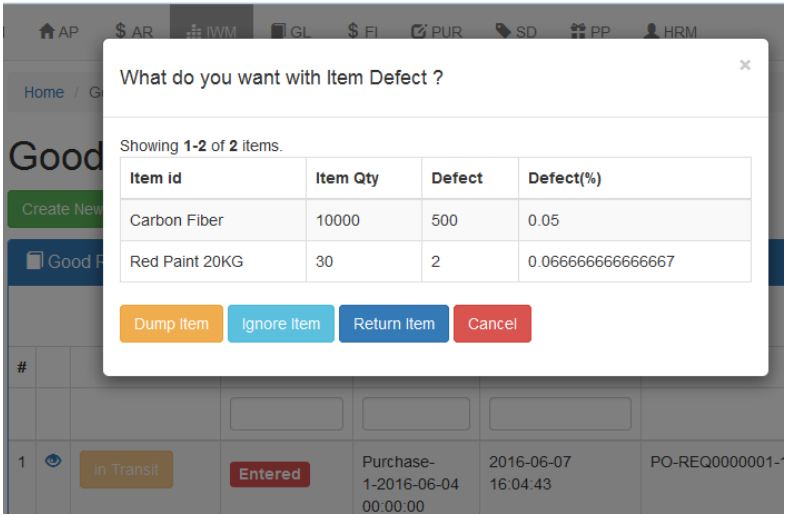
Gambar A.2 View Create Warehouse



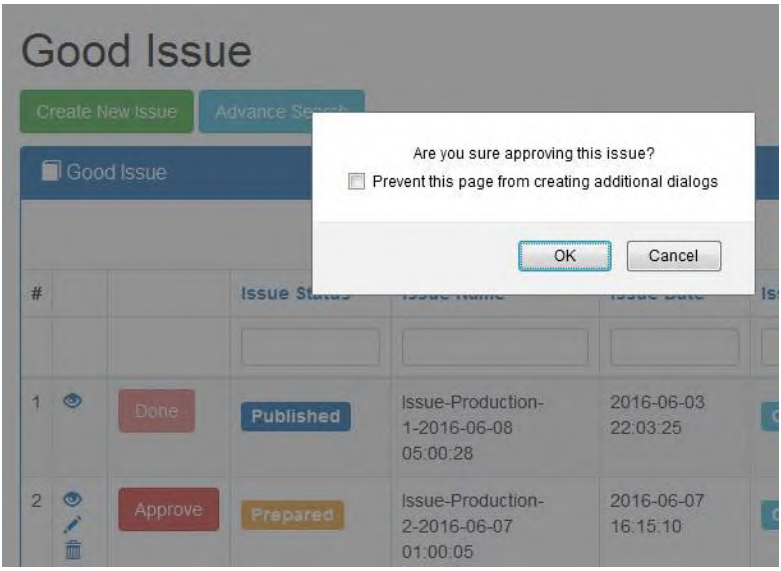
Gambar A.3 View Update Warehouse



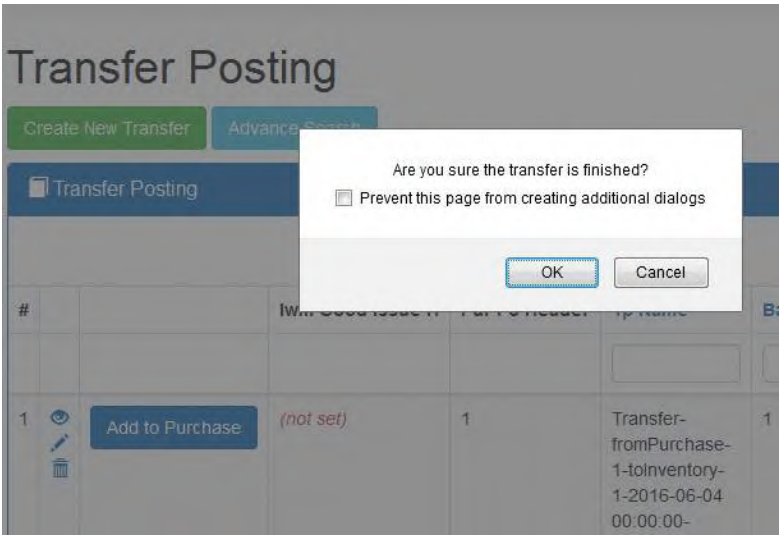
Gambar A.4 View Delete Warehouse



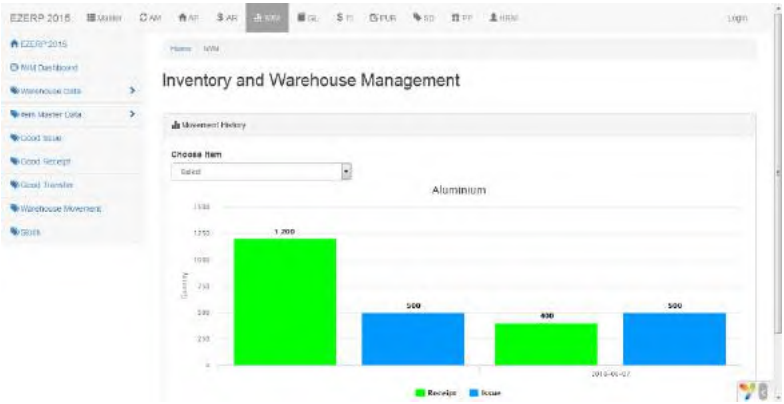
Gambar A.5 View Process Good Receipt



Gambar A.6 View Process Good Issue

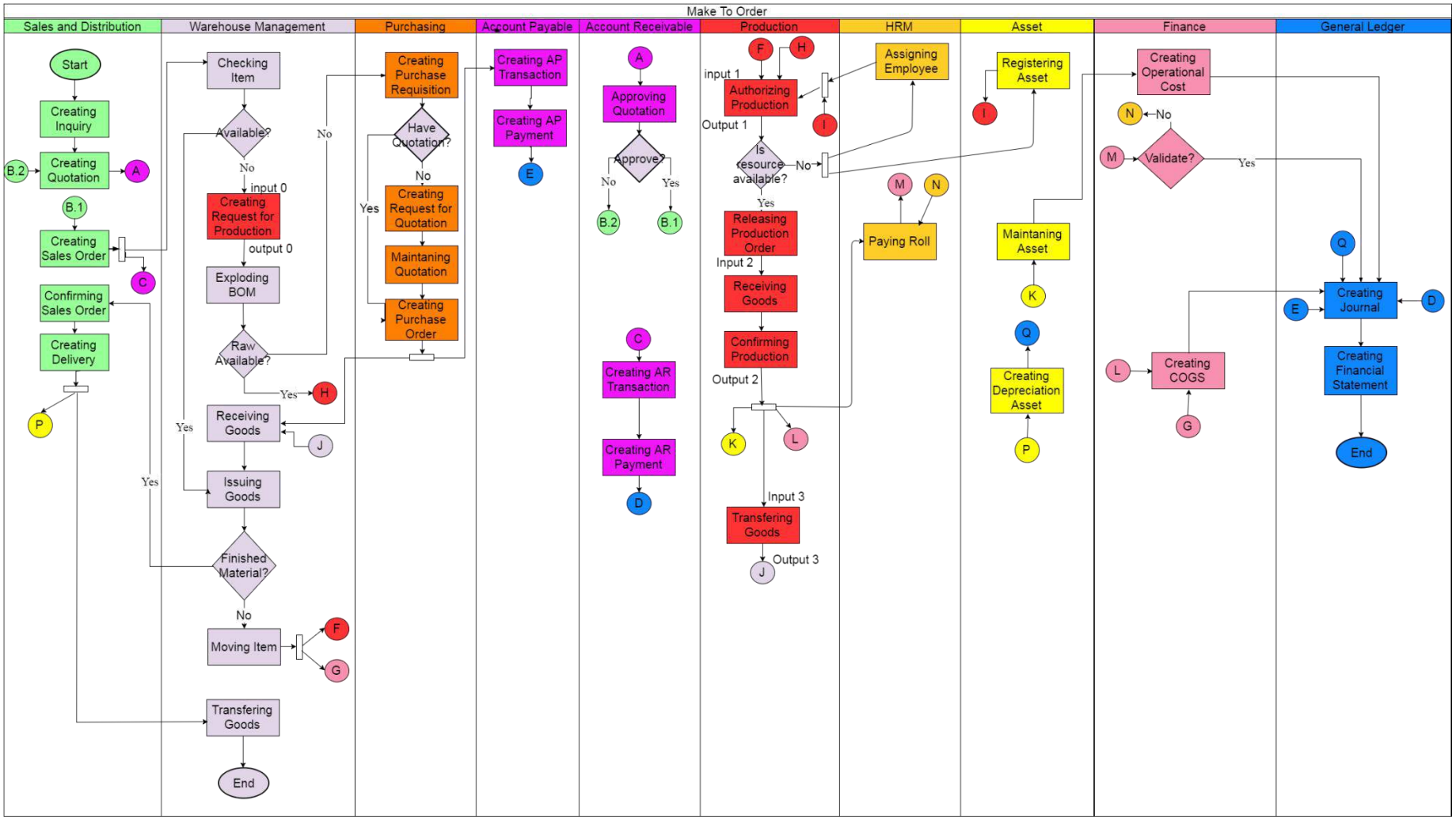


Gambar A.7 View Process Transfer Posting

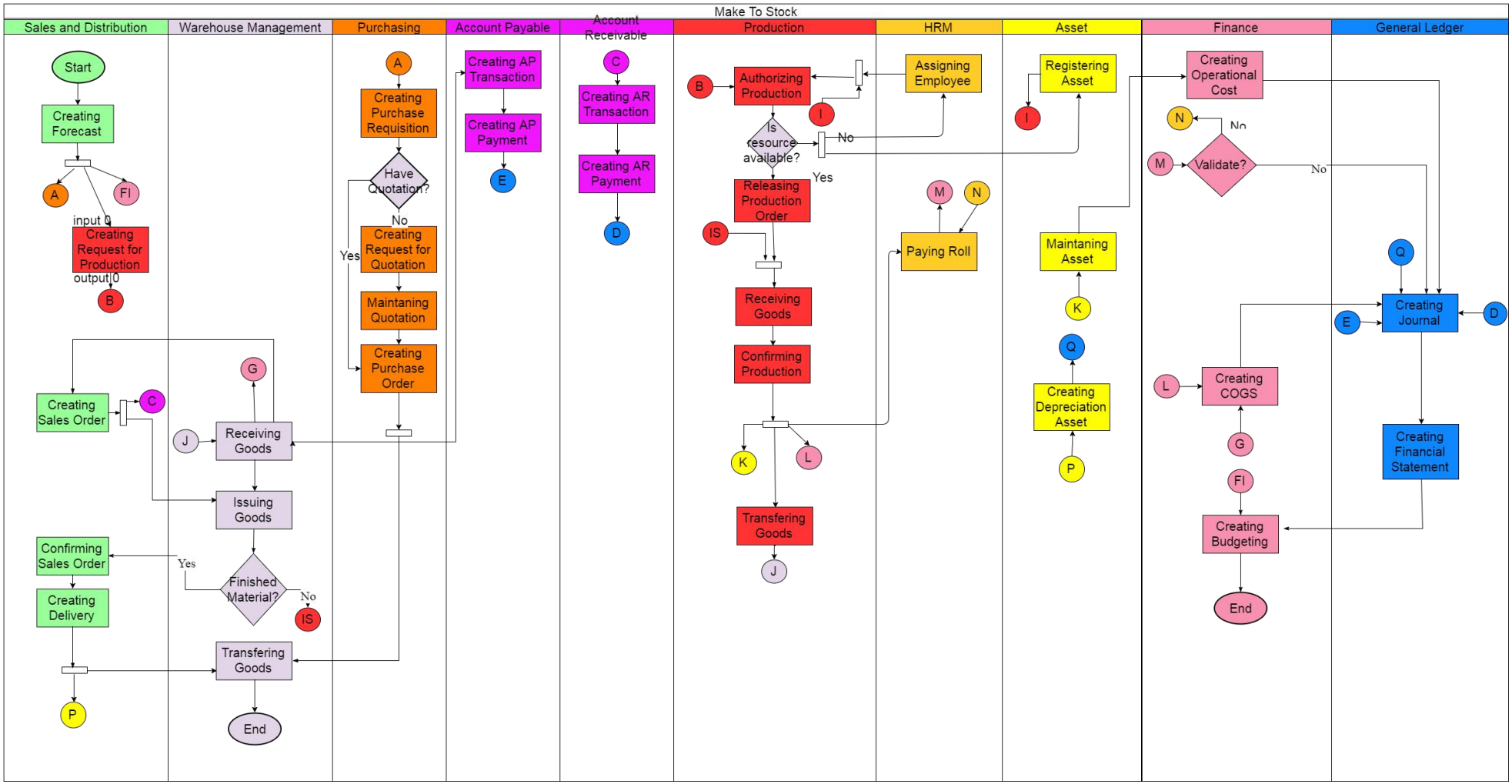


Gambar A.8 Halaman Utama IWM

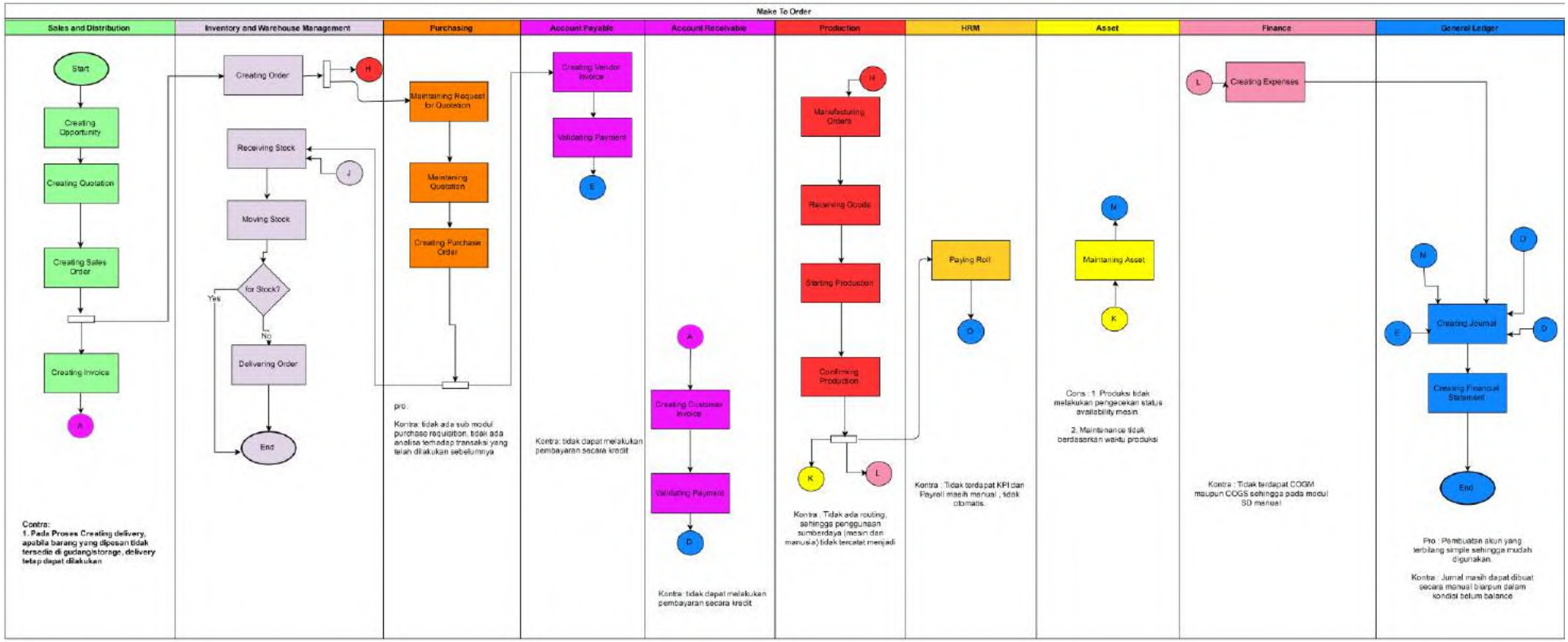
Bisnis Proses



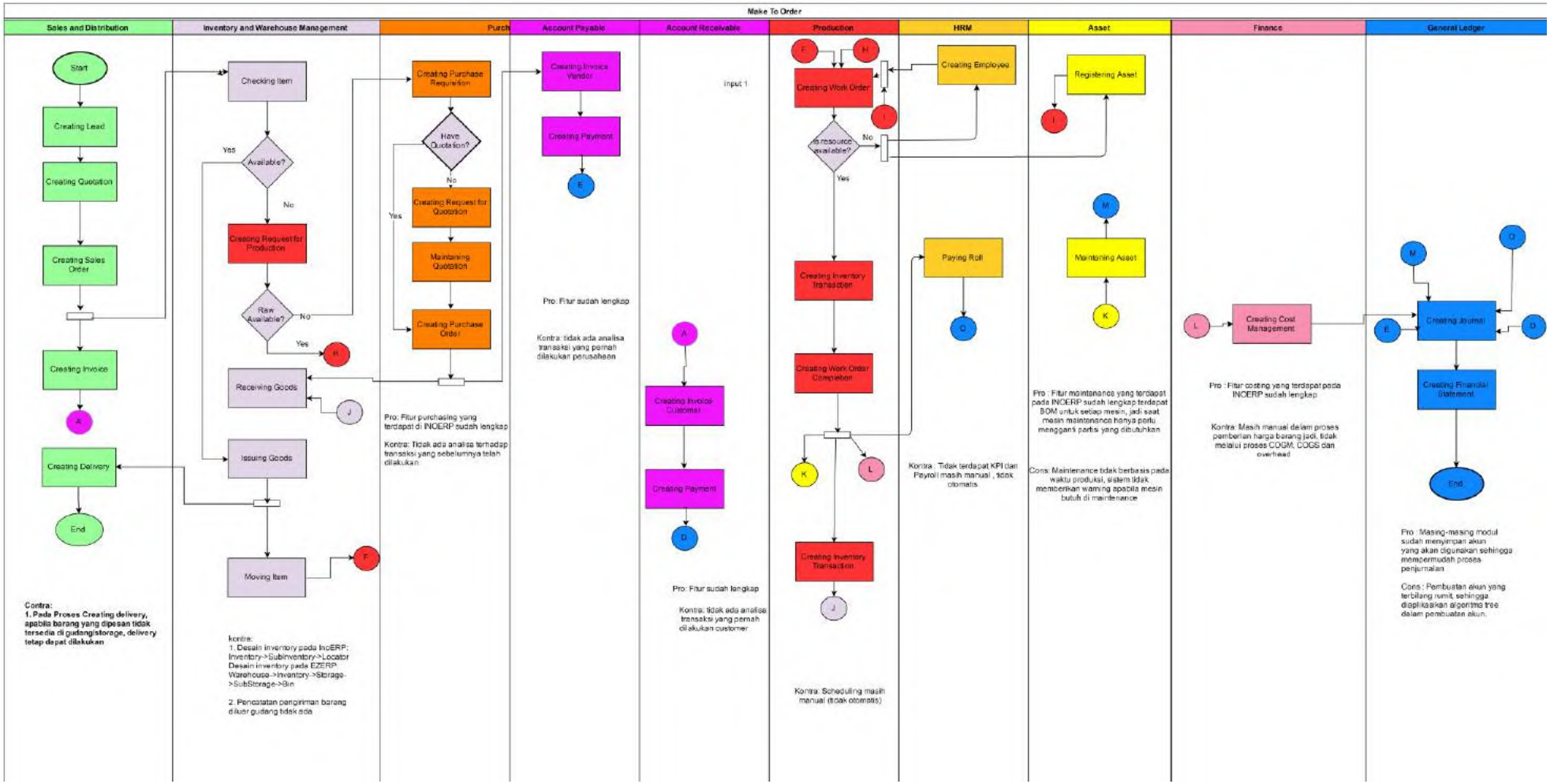
Gambar B.9 Bisnis Proses MTO (*Make To Order*)



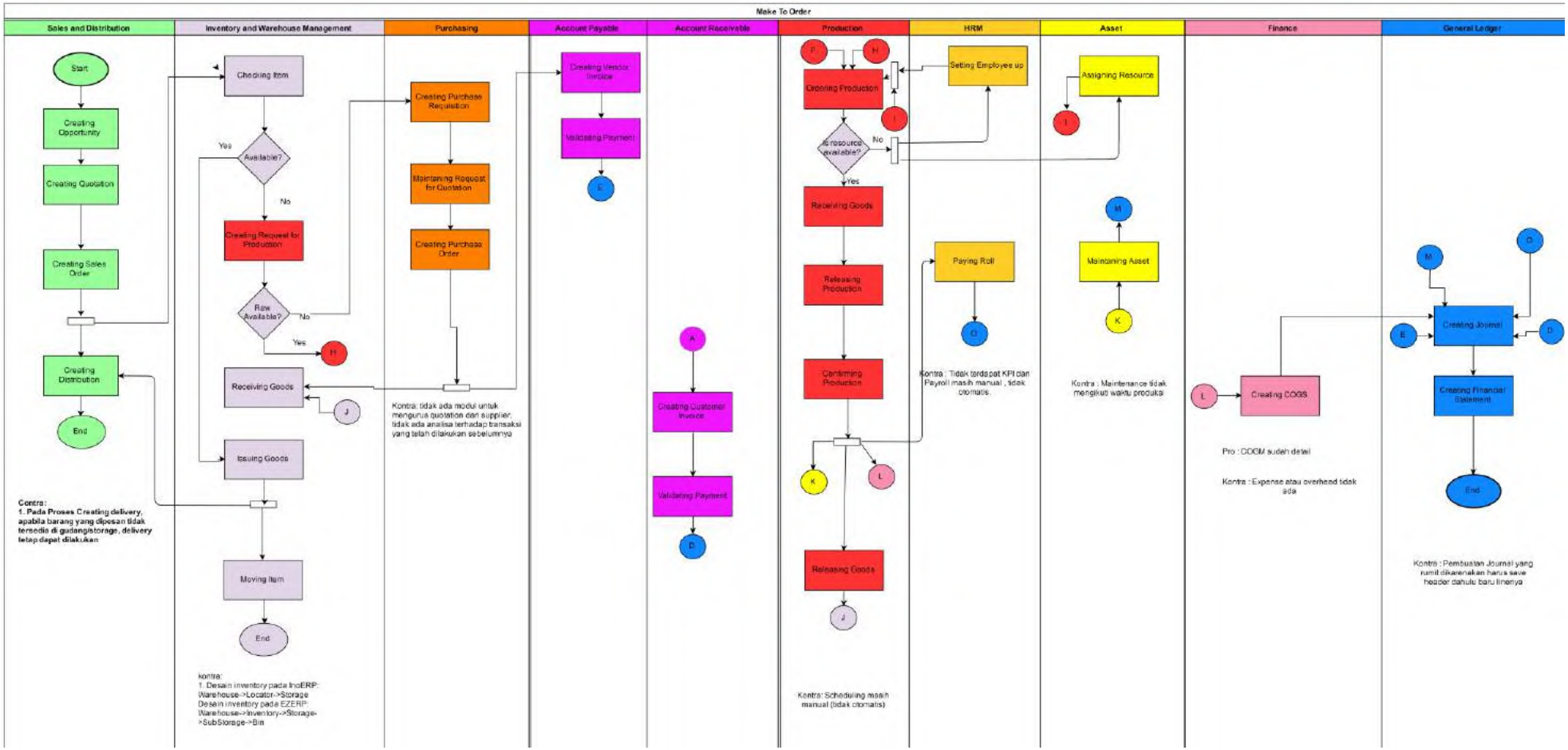
Gambar B.10 Bisnis Proses MTS (*Make To Stock*)



Gambar B.11 Bisnis Proses Odoo



Gambar B.12 Bisnis Proses InoERP



Gambar B.13 Bisnis Proses Adempiere

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan Tugas Akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Memodelkan dengan mencatat desain *warehouse* hingga detail dapat memproses data sistem dengan lebih jelas khususnya dalam pencatatan transaksi yang terdapat pada gudang. Proses *trace* pada *warehouse* dapat dengan mudah dilakukan.
2. Menggabungkan data pada catatan pembelian/produksi dengan catatan item master dapat mengimplementasikan konsep FIFO beserta penggabungan data tersebut dengan Bin yang disebut dengan *Quant* dapat mengimplementasikan proses peletakan barang secara FIFO pada *warehouse.*
3. Menggunakan lebih dari satu *database server* dan menggunakan teknik replikasi pada *distributed database* mampu mengatasi kegagalan sistem yang disebabkan oleh *database server.*
4. Menggunakan 4 server dimana 1 *server* bertindak sebagai *server* aplikasi, 2 *server* merupakan *database server* dan satu *server* bertindak sebagai *management server* yang bertugas dalam memproses replikasi terhadap kedua *database server.*

6.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Melakukan penambahan proses bisnis dalam mengatasi barang yang rusak (*Defect*).

2. Menambahkan atau memperbaiki proses pencatatan pada *warehouse* dengan menambahkan aspek *user experience* untuk mempermudah dalam proses pengelolaan *warehouse.*
3. Menambahkan proses bisnis dalam pengelolaan *inventory* khususnya dalam pengelolaan gudang yang dalam tahap perbaikan atau pengembangan.
4. Menambahkan atau memperbaiki metode yang sudah ada untuk lebih meningkatkan kinerja *inventory.* Dengan menggabungkan teknik FIFO, LIFO dan FEFO.
5. Meningkatkan *running process* untuk proses dengan data yang sangat besar.

DAFTAR PUSTAKA

[1] Simha R. Magal, Jeffrey Word, Integrated Business Processes with ERP System, United States of America: John Wiley and Sons, Inc., 2012.

[2] Riswandy Setiaputra Godlieb, Prof. Drs. Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D., Dwi Sunaryono, S.Kom., M.Kom., Rancang Bangun Aplikasi Berorientasi Arsitektur Service (SOA) dengan Pendekatan Workflow pada Domain Inventory Enterprise Resource Planning, Surabaya, 2013.

[3] Y. Sutanto and R. Sarno, "Inventory management optimization model with database synchronization through internet network (A simulation study)," pp. 115-120, 2015.

[4] J. Salvo, P. Mackenzie, J. Bennett, H. Relyea and A. Thomas, "Inventory management system and method". United States Patent 6,341,271, January 2002.

[5] T.Cheng, J.Yue and T.Guo, "Inventory Modeling in Supply Chain Management: A Review," pp. 1-4, 2008.

[6] R. P. Derstine and R. J. Huefner, "LIFO-FIFO, Accounting Ratios and Market Risk," *Journa of Accounting Research,* vol. 12, pp. 216-234, 1974.

[7] H. G. Hunt, "Potential Determinants of Corporate Inventory Accounting Decisions," *Journal of Accounting Research,* vol. 23, pp. 448-67.

[8] F. W. Lindahl, "Dynamic Analysis of Inventory Accounting Choice," *Journal of Accounting Research,* vol. 27, pp. 201-26, 1989.

[9] L. Ozdamar and M. A. Ertem, "Models, solutions and enabling technologies in humanitarian logistics," *European Journal of Operational Research,* vol. 244, no. 1, pp. 55-65, 2015.

[10] A. Dan and D. Towsley, "An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes," *SIGMETRICS Perform. Eval. Rev*, vol. 18, pp. 143-152, 1990.

[11] R. Sarno, C. Djeni, I. Mukhlash and D. Sunaryono, "Developing A Workflow Management System for Enterprise Resource Planning," *Journal of Theoretical and Applied Information Technology*, vol. 72, pp. 412-421, 2015.

[12] N. Dopuch and M. Pincus, "Evidence on the Choice of Inventory Accounting Methods: LIFO Versus FIFO," *Journal of Accounting Research*, vol. 26, pp. 28-59, 1988.

[13] D. Morse and G. Richardson, "The LIFO/FIFO Decision," *Journal of Accounting Research*, vol. 21, pp. 106-127, 1983.

[14] D. C. Yen, D. C. Chou and J. Chang, "A synergic analysis for Web-Based enterprise resource planning systems," *Computer Standards & Interfaces*, vol. 24, no. 4, pp. 337-346, 2002.

[15] R. Sarno and Heryanti, "A Service Portofolio for an Enterprise Resource Planning," *International Journal of Computer Science and Network Security*, vol. 10, pp. 144-156, 2010.

INDEX

BIODATA PENULIS

I Gede Arya Putra Perdana atau yang biasa disapa dengan nama Arya atau Dede, lahir di Denpasar pada tanggal 1 April 1994. Memiliki seorang adik perempuan dan telah menempuh pendidikan di SD Negeri 17 Kesiman (2000-2006), SMP Negeri 3 Denpasar (2006-2009), SMA Negeri 8 Denpasar (2009-2012) dan saat ini sedang menempuh pendidikan Sarjana di Teknik Informatika, Fakultas Tenologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya angkatan tahun 2012.

Memiliki ketertarikan dalam segala bentuk kegiatan sosial dan pernah menjadi asisten dalam matakuliah Pemrograman Berorientasi Objek. Terlibat aktif dalam organisasi kemahasiswaan serta kepanitiaan selama perkuliahan, antara lain staff Pengembangan Sumber Daya Mahasiswa di Himpunan Mahasiswa Teknik Computer-Informatika ITS 2013/2014, Staff Hubungan Masyarakat acara Schematics 2013, Staff Hubungan Masyarakat acara Schematics 2014, Staff Pengabdian Masyarakat TPKH-ITS (Tim Pembina Kerohanian Hindu) 2013/2014. Steering Committee acara Simakrama dan Upanayana 2015. Untuk komunikasi, dapat dihubungi melalui surel : ashura.shinji@gmail.com